Decoding Strategies for Syntax-based Statistical Machine Translation

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Fabienne Braune

aus Thun/Schweiz

Hauptberichter:	Dr. Andreas Maletti
Mitberichter 1:	Dr. Alexander Fraser
Mitberichter 2:	Prof. Dr. Jonas Kuhn
Mitberichter 3:	Prof. Dr. Kevin Knight

Tag der mündlichen Prüfung: 30.11.2015

Institut für Maschinelle Sprachverarbeitung (IMS)
Universität Stuttgart

2015

Abstract

Translation is the task of transforming text from a given language into another. Provided with a sentence in an input language, a human translator produces a sentence in the desired target language. The advances in artificial intelligence in the 1950s led to the idea of using machines instead of humans to generate translations. Based on this idea, the field of *Machine Translation* (MT) was created. The first MT systems aimed to map input text into the target translation through the application of hand-crafted rules. While this approach worked well for specific language-pairs on restricted fields, it was hardly extendable to new languages and domains because of the huge amount of human effort necessary to create new translation rules. The increase of computational power enabled Statistical Machine Translation (SMT) in the late 1980s, which addressed this problem by learning translation units automatically from large text collections.

Statistical machine translation systems can be divided into several paradigms depending on the form of the (automatically learned) units used during translation. Early systems modeled translation between words. Later work extended these units from single words to sequences of words called *phrases*. A common point between word and phrase-based SMT is that the translation process takes place sequentially. This left-to-right process is not well suited to translate between languages where several words need to be reordered over (potentially) long distance. Such reorderings, which take place between many language pairs (e.g. English-German, English-Chinese or English-Arabic), led to the implementation of SMT systems based on formalisms that allow to translate *recursively* instead of sequentially. In these systems, called *syntax-based* systems, the (automatically learned) translation units are modeled with formal grammar productions and translation is performed by assembling the productions of these grammars.

Many different grammar formalisms have been developed to model translation. One of the first is the Synchronous Context-Free Grammar (SCFG) which is an extension of the well-known Context-Free Grammar (CFG). To overcome several drawbacks of SCFG, more powerful formalisms have been explored such as the

Synchronous Tree Substitution Grammar (STSG) or the local Multi Bottom-Up Tree Transducer (l-MBOT). Because formal grammars can encode information in their non-terminal symbols, linguistic annotations can easily be integrated into syntax-based systems. Such annotations have been integrated at several levels. One of the first large scale and high performance approaches is the *hierarchical* system which uses SCFG rules without linguistic annotations. Another high-performance system uses STSG rules with linguistic information on the target language.

This thesis contributes to the field of syntax-based SMT in three ways. First, the applicability of a new grammar formalism to SMT is tested by building the first system based on the l-MBOT. Previous to this work, procedures to automatically learn l-MBOT translation rules were developed. However, the implementation of the translation process and an empirical evaluation remained to be done. Our work closes this gap.

The second contribution is the exploration of new ways to integrate linguistic annotations in l-MBOT based systems. Three systems have been implemented that work with annotations at different levels. A first variant works without any annotations. A second variant uses annotations of the input and target languages and a third one integrates target annotations only.

The last contribution explores new ways to integrate linguistic annotations in the translation model instead of encoding those in the translation rules. In this work, a hierarchical system is augmented with syntactic annotation in the form of soft syntactic features. To this end, a discriminative model has been defined and trained, which refines rule selection by taking the syntactic structure of the source sentence into account. This model is added to the hierarchical translation model as an additional feature. This model has been extended to also work on systems integrating annotations of the target language.

Deutsche Zusammenfassung

Übersetzung ist der Prozess, Texte von einer Sprache in eine andere zu transformieren. Sätze aus einer Sprache werden durch einen menschlichen Übersetzer in die Zielsprache überführt. Die Fortschritte in Künstlicher Intelligenz in den 1950er Jahren haben dazu geführt, dass ab diesem Zeitpunkt ebenfalls Computer für die Übersetzung eingesetzt wurden. Dies ist die Geburtsstunde der Maschinellen Übersetzung (MÜ). Die ersten MÜ-Systeme basierten auf handgeschriebenen Regeln, die Texte aus einer Sprache auf eine andere Sprache abbilden können. Dieser Ansatz eignet sich zwar sehr gut für einige Sprachpaare in gewissen Anwendungsbereichen, kann aber nicht ohne sehr kostspieliges Regelschreiben für neue Sprachen oder Anwendungen angepasst werden. Die Verfügbarkeit höherer Rechenleistung hat in den späten 1980er Jahren dazu geführt, dass sich Statistische Maschinelle Übersetzung (SMÜ) etablieren konnte. SMÜ-Systeme lernen die nötigen Übersetzungseinheiten automatisch aus großen Textsammlungen.

SMÜ-Paradigma können abhängig von ihrer automatisch gelernten Übersetzungseinheiten in verschiedene Paradigmen eingeteilt werden. Die ersten Systeme modellierten nur die Übersetzung von einzelnen Wörtern. Spätere Systeme erweiterten dies zu Wortsequenzen, welche als Phrasen bezeichnet werden. Eine Gemeinsamkeit beider Ansätze besteht darin, dass die Übersetzung sequentiell erfolgt. Dieser von-Links-nach-Rechts Ansatz ist nicht sehr praktikabel, falls Wörter im Rahmen der Übersetzung über eine lange Distanz im Satz neu geordnet werden müssen. Typische Sprachpaare, die dieses Phänomen aufweisen, sind Englisch-Deutsch, Englisch-Chinesisch oder Englisch-Arabisch. Um diesem Problem entgegenzuwirken, wurde ein neues SMÜ Paradigma eingeführt, welches die Übersetzung rekursiv statt sequentiell durchführt. Diese sogenannten syntax-basierten SMÜ Systeme basieren auf Übersetzungseinheiten, die durch Produktionen einer formalen Grammatik dargestellt werden.

Viele unterschiedliche Grammatikformalismen wurden zur Modellierung von Übersetzung entwickelt. Einer der ersten waren synchrone kontextfreie Grammatiken (SCFG), welche eine Erweiterung der kontextfreien Grammatiken

darstellen. Um einige Nachteile der SCFGs auszugleichen, wurden mächtigere Formalismen untersucht. Darunter fallen Baumsubstitutionsgrammatiken (STSG) oder der lokale, aufsteigende Mehrfachbaumübersetzer (l-MBOT). Da formale Grammatiken Informationen in ihren Nicht-Terminalen speichern können, lassen sich linguistische Annotationen sehr einfach über diese Symbole in den Formalismus übertragen. Solche Annotationen können auf unterschiedliche Arten eingeführt werden. Das erste System, das im großen Maßstab eingesetzt wurde, ist das hierarchische Modell. Dieses verwendet SCFG-Regeln, die als Nicht-Terminale nur ein generisches Symbol verwenden. Ein anderer erfolgreicher Ansatz sind Systeme mit linguistischen Annotationen aus der Zielsprache.

Diese Arbeit trägt zu dem Gebiet der syntax-basierten SMÜ in dreierlei Hinsicht bei. Erstens wurde ein neuer Grammatikformalismus basierend auf l-MBOT implementiert und getestet. Dieser Arbeit ging eine Vorarbeit zur automatischen Extraktion von l-MBOT-Grammatiken voraus. Jedoch gab es bisher weder einen Übersetzungsalgorithmus noch eine empirische Evaluation dieses Modells. Diese Arbeit schliesst diese Lücke.

Als Zweites wurden neue Möglichkeiten untersucht, wie linguistische Annotationen in einem l-MBOT-basierten MÜ System eingesetzt werden können. Dazu wurden drei Systeme implementiert, die verschiedene Annotationen verwenden. Die erste Variante verwendet keine Annotation, die zweite Variante betrachtet Annotationen für die Quell- und die Zielsprache. In der letzten Variante werden nur Annotationen in der Zielsprache betrachtet.

Abschließend wurde untersucht, wie linguistische Annotationen im Übersetzungsmodell verankert werden können, anstatt sie wie herkömmlich über die Verarbeitung der Übersetzungsregeln zu betrachten. Dazu wurde ein hierarchisches System mit syntaktischen Annotationen angereichert. Dies konnte durch ein diskriminatives Modell, welches als zusätzliches Merkmal im hierarchischen MÜ System integriert ist, erreicht werden. Dieses Modell erlaubt eine bessere Regelauswahl durch das Betrachten der syntaktischen Analyse des Quellsatzes. Darüber hinaus wurde das Modell erweitert, damit ebenfalls Annotationen der Zielsprache während der Übersetzung einbezogen werden können.

Contents

Chapter 1

Introduction

1.1 Syntax-based SMT

The goal of translation is to transform text from a given language into another. Given the English sentence E below, a human translator produces a sentence in a required target language such as, for instance, the German sentence G.

E *The commission has accepted the proposition.*

G *Die Kommission hat den Vorschlag angenommen.*

The advances in artificial intelligence in the 1950s led to the idea of using computer programs instead of humans to perform translation tasks. The development of this idea created the field of *machine translation*.

The first machine translation systems aimed to map input text into the target translation through the application of hand-crafted rules. While early systems used simple rules, later frameworks integrated more sophisticated translation units, which included rich linguistic knowledge such as morphology, syntax or semantics. Although this (rule-based) approach worked well for specific language-pairs on restricted fields, it could not easily be extended to new languages and domains due to the huge amount of human efforts necessary to create new translation rules. With the increase of computing power, a major breakthrough in the field took place in

the late 1980s with the application of statistical methods to machine translation. Through statistical models, translation units could be automatically learned from large text collections. Consequently, Statistical Machine Translation (SMT) was not tied anymore to specific languages or domains.

SMT systems can be divided into several paradigms depending on the form of the (automatically learned) units used during translation. Early systems [Brown et al., 1990, Brown et al., 1993] modeled translation between words such as, for instance, the translation of *forecasts* into *Prognosen* in Figure 1.1.

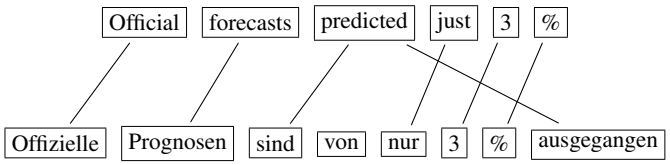


Figure 1.1: English-to-German Translation. The links indicate correspondences between the **words** in these sentences.

Later work [Koehn et al., 2003] extended these units from single words to sequences of words called *phrases*. The translation of *Official forecasts* into *Offizielle Prognosen* in Figure 1.2 shows an example phrase-pair.

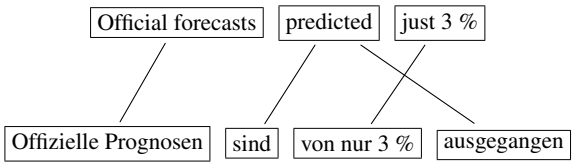


Figure 1.2: English-to-German Translation. The links indicate correspondences between **phrases** in these sentences.

A common point between word and phrase-based SMT is that the translation process works sequentially. The main drawback of this left-to-right method is that it often fails to correctly reorder words, especially over long distances. For instance, the translation of the English verb *predicted* into the German *sind ausgegangen* and its reordering are difficult to obtain using sequential systems.

But many language pairs (e.g. English-Chinese, English-Arabic, English-Japanese, and English-German) require to reorder words over long distances. Even in language pairs such as English-French, which have a similar word order, translation in specific domains requires complex reordering. Figure 1.3 shows an example French-English translation in the scientific domain.

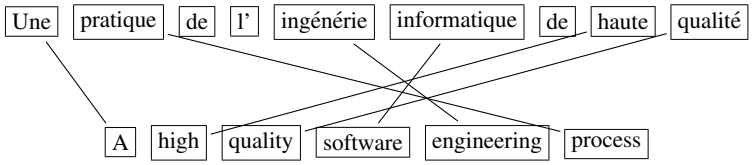


Figure 1.3: French-to-English Translation

In order to deal with these word reorderings, many authors proposed to build SMT systems based on formalisms that allow to translate sentences recursively instead of sequentially. These systems use formal grammars to model translation and are called *syntax-based* systems.

While many approaches have been proposed to build syntax-based SMT systems roughly at the same time (an overview is given in Section 2.2), one of the first large scale and high performance approach is the hierarchical phrase-based model in [Chiang, 2005]. In this system, the translation units are rules such as p_1 to p_3 below, which are composed of phrases with an additional symbol X.

p_1 $X \rightarrow \langle\ X_1$ predicted X_2, X_1 sind von X_2 ausgegangen $\rangle$

p_2 $X \rightarrow \langle$ just 3 %, nur 3 %, $\rangle$

p_3 $X \rightarrow \langle$ Official forecasts, Offizielle Prognosen $\rangle$

Translation with these rules is done recursively by replacing the symbols labeled by X with further rules.[1] For instance, to translate the German sentence in Figure 1.1, rule p_1 would be applied first, followed by p_2 and p_3. The advantage of this recursive mechanism is that reordering is encoded in the translation rules. For instance, rule p_1 allows to simultaneously (i) translate the verb *predicted* into *sind ausgegangen* and (ii) indicate that *ausgegangen* will be separated from *predicted* by a gap.

[1] More precisely, indices on the symbol X indicate where to plug the replacing rules.

With this improved reordering procedure, hierarchical systems have outperformed sequential frameworks on several translation tasks involving language pairs with many (potentially long-distance) reorderings such as English and Chinese or Arabic. The theoretical foundations of the hierarchical model are Synchronous Context-Free Grammars (SCFG) [Aho and Ullman, 1969].[2] In particular, hierarchical grammars are SCFG where X is the only non-terminal. While this absence of annotations makes the hierarchical approach very flexible, it also enables rule applications that lead to malformed translations.

Instead of using just X, SCFG rules can be labeled by any finite alphabet such as, for instance, a set of linguistic syntactic annotations. The parse trees in Figure 1.4 show example syntactic annotations for the English and German sentences presented above. In order to use these annotations to guide the translation process, several approaches, discussed in Section 2.2.1, integrated those in SMT systems by using annotated SCFG rules. In these approaches, the input to the translation process is a sentence together with a parse tree, such as the English sentence in Figure 1.4. During translation, rules matching the labels of this tree are recursively applied until a corresponding German tree is generated. The concatenation of its leaves yields the obtained target translation.

Unfortunately, SCFG-based systems carrying syntactic annotations have performed poorly compared to the (non-annotated) hierarchical system.[3] The main reason for this failure is that the annotation-driven translation process, which requires (i) to match input parse labels and (ii) to assemble rules in a target tree, is very restrictive and error prone [Ambati and Lavie, 2008, Ambati et al., 2009]. Three main strategies have been adopted to take advantage of linguistic annotations without decreasing the performance of syntax-based systems.

The first switches from SCFG to more powerful formalisms such as Synchronous Tree Substitution Grammars [Eisner, 2003] (STSG), which model translation rules

[2] A detailed presentation of synchronous grammars is given in Chapter 2.1.

[3] SCFG-based systems with (source and target) syntactic annotations also underperformed sequential systems even on language pairs with many reorderings such as English and Chinese.

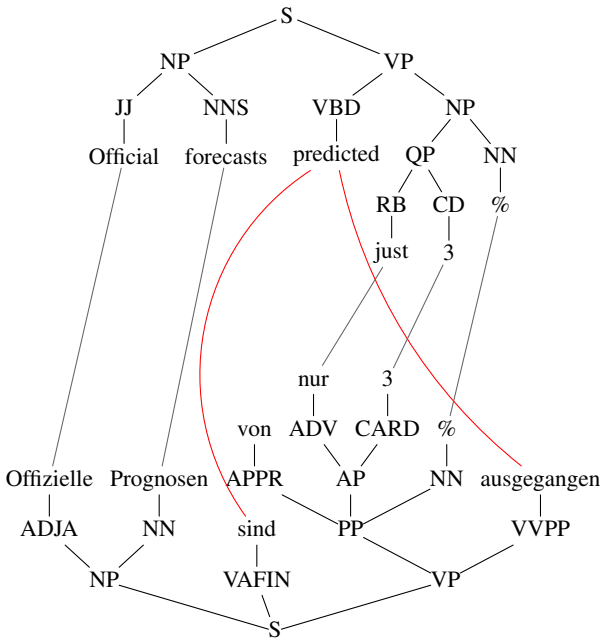


Figure 1.4: Word aligned biparsed sentence pair

as pairs of trees instead of strings containing non-terminals. Figure 1.5 shows an example STSG rule capturing the translation of the verb *predicted* into *sind ausgegangen*. By modeling complex reorderings[4], STSG-based systems have outperformed their SCFG-based counterpart. Further work extended STSG into even more powerful formalisms such as Synchronous Tree Sequence Substitution Grammars (STSSG) [Zhang et al., 2008, Sun et al., 2009] which work with sequences of trees instead of single ones. We contribute to this research by implementing a system based on the Local Multi-Bottom Up Tree Transducers (l-MBOT) [Maletti, 2011], a novel formalism that offers a middle-ground between STSG and STSSG.

The second strategy reduces the amount of annotations in the translation rules. Instead of working with fully annotated rules, several approaches (presented in Section 2.2) drop the source or target side annotations. Among these, systems keeping the target labels only have been high ranked in public evaluation campaigns [Bojar et al., 2014]. This success can be explained by the fact that the

[4]Such as, for instance, the swapping of non-terminals that are at different levels in a tree.

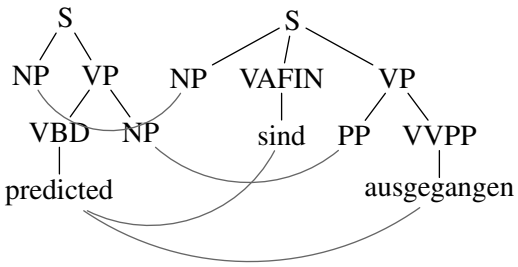


Figure 1.5: Example STSG rule

absence of source annotations provides high flexibility while the target labels guarantee the syntactic well-formedness of the output. Following this work, we build l-MBOT-based systems with linguistic annotations at different levels and show that systems with target annotations only perform best.

The third strategy, presented in Section 3, integrates linguistic information in the translation model of a hierarchical system instead of encoding it in the grammar rules. In this work, soft syntactic constraints guide the correct application of non-annotated rules. This methods allows to take advantage of linguistic information while keeping the flexibility of the hierarchical model. We make two contributions to this field. First, we propose a novel way to implement soft syntactic constraints by integrating a global rule selection model in a hierarchical system. Second, we define and evaluate the first rule selection model for systems with target syntactic annotations.

This thesis has two goals. The first is to present our contributions to the field of syntax-based SMT. The second is to provide a consistent and detailed presentation of topics that have not been described elsewhere. For instance, in the field of formal grammars for SMT, [Chiang, 2006] gives an excellent overview of synchronous grammars. However, this description does not include recent formalisms such as STSSG or l-MBOT. We close this gap by providing a detailed description of these formalisms in Section 2.1. We also provide a complete presentation of decoding procedures for SCFG grammars. Although these are well presented in

[Chiang, 2007] and [Hoang, 2011] there is no complete overview of these strategies for SCFG grammars with syntactic annotations. We provide a complete description of decoding for SCFG grammars in Chapter 4.

1.2 Contributions

1.2.1 Theoretical Contributions

We make three contributions to the field of syntax-based SMT:

- **The first SMT system based on Local Multi Bottom-Up Tree Transducers**: We build an SMT system using the Local Multi Bottom-Up Tree Transducer as translation model (l-MBOT). Our model includes features specific to l-MBOT rules such as a gap penalty that counts the number of elements in the discontiguous target sides. Decoding is done with an extended bottom-up chart parser that can generate partial translations made of discontiguous units. Language model scoring is integrated in the decoding process and pruning is applied to reduce the computing costs. We show that with certain levels of linguistic annotations our system outperforms several baselines based on synchronous context-free grammars. A detailed overview of this contribution is given in Section 2.3. The complete contribution is presented in Chapter 5.

- **A global and exhaustive rule selection model for hierarchical SMT**: While previous work on hierarchical rule selection build models that are either local to the source side of the translation rules or heavily pruned, we propose a global model that performs no pruning. Because it generalizes to the complete hierarchical grammar and is not pruned, our global model captures useful information that is lost in local and pruned approaches. We show in an extensive evaluation that a hierarchical system using this additional information outperforms systems integrating local and pruned models. A detailed

overview of this contribution is given in Section 3.4. The complete contribution is presented in Chapter 6.

- **The first rule selection model for SMT with target syntactic annotations**: We extend our global rule selection model to work on systems with syntactic annotations on the target language side, also called *string-to-tree systems*. This contribution is the first attempt to use rule selection in string-to-tree systems. A preliminary evaluation shows that rule selection does not improve string-to-tree SMT. A detailed overview of this contribution is given in Section 3.4. The complete contribution is presented in Chapter 7.

1.2.2 Software Contributions

Our theoretical contributions are implemented in the Moses toolkit [Koehn et al., 2007, Hoang et al., 2009]:

- Our l-MBOT system is implemented in the branch `mbotTestedDecoder` and can be downloaded with the command `git clone -b mbotTestedDecoder https://github.com/moses-smt/mosesdecoder.git`
- Our rule selection models are implemented in the branch `syntaxContext` and can be downloaded with the command `git clone -b syntaxContext https://github.com/moses-smt/mosesdecoder.git`

1.3 Outline of the Dissertation

In this Section, we present the outline of this thesis. We present each contribution listed above (Section 1.2) and discuss the parts of the thesis related to each contribution. Then we outline each contribution separately.

Our first contribution is in the field of formal grammars and their implementation into SMT systems, which we present in Chapter 2. We begin this chapter with

a detailed presentation of synchronous grammars in Section 2.1, which leads to the definition of l-MBOT (Section 2.1.3). The aim of this presentation is to provide the reader with background information necessary to understand the inner workings of l-MBOT. It also gives a coherent presentation of grammar formalisms including recent grammars such as Synchronous Tree Sequence Substitution Grammars (STSSG). In Section 2.2, we present SMT systems based on the presented grammars, the field to which our work contributes. We conclude the chapter by giving an overview of this contribution in Section 2.3.

Our second and third contributions are on the topic of soft syntactic constraints and their integration into SMT systems. We present this topic in Chapter 3. We begin by showing why hierarchical systems can benefit from soft syntactic constraints in Section 3.1. Then we discuss several ways of integrating these constraints into syntax-based SMT systems, which have been presented in previous work (Section 3.2). We close the chapter with a detailed overview of our contributions in Section 3.4.

A central aspect of our contributions is the design and implementation of customized decoding procedures. These are basically extensions of the algorithms used to decode with Synchronous Context-Free Grammars (SCFG). In Chapter 4 we provide the reader with background knowledge in decoding strategies for SCGF-based SMT. We also give a brief overview of existing algorithms to obtain SCFG grammars from bilingual texts. In Section 4.1, we present hierarchical systems, i.e. without syntactic annotations. In Section 4.3 we present systems including syntactic annotations. This chapter offers a coherent overview of decoding procedures for SCFG-based systems as implemented in the Moses toolkit [Hoang et al., 2009].

In Chapter 5 we present our SMT system based on l-MBOT. We work on a restricted form of l-MBOT, which uses shallow rules (Sh-l-MBOT). After a brief presentation of existing rule extraction procedures in Section 5.1, we present our contributions. We propose a translation model that defines and trains features specific to Sh-l-MBOT rules (Section 5.2) before presenting a decoding procedure for this model (Sections 5.3 and 5.4). Our system works with Sh-l-MBOT without syntactic

annotations or systems containing annotations at different levels. An evaluation of our system is given in Section 5.5.

In Chapter 6 we present our global rule selection model for hierarchical SMT. In Sections 6.2 and 6.3 we formulate the model and describe the training procedure. The integration of this model in a hierarchical system requires to modify the decoder in several ways, which we present in Section 6.4. We discuss the advantages of our approach in Section 6.5 before giving an extensive evaluation in Section 6.6.

In Chapter 7 we apply our hierarchical rule selection model to systems with target syntactic annotations (also called *string-to-tree* systems). We begin by showing that in order to work with the string-to-tree system implemented in the Moses toolkit [Hoang et al., 2009] our model has to be redefined (Section 7.1). In Section 7.2 present the formulation and training procedure of the adapted model. We evaluate our approach in Section 7.3.

In Chapter 8 we discuss again our contributions and present shortcomings of our work. We close this thesis with a discussion of future work.

Chapter 2

Synchronous Grammars and their Implementation

Syntax-based Statistical Machine Translation (SMT) systems find their theoretical foundations in the field of formal languages. In the same fashion as statistical syntactic parsing, syntax-based SMT encodes the structure of natural language into grammars. While parsing works on monolingual data and hence models string or tree generation, machine translation needs to capture the relation between language pairs. Many studies have extended grammar formalisms such as Context-Free Grammars (CFG) [Hopcroft et al., 2006] or Tree Substitution Grammars (TSG) [Eisner, 2003] to model the generation of string and tree *pairs*. These formalisms are often referred to as *synchronous grammars*. Based on these models, different syntax-based SMT systems have been presented in the literature.

Recent work in the field of formal languages introduced new formalisms such as the Local Multi Bottom-up Tree Transducer (l-MBOT) [Maletti, 2010]. Subsequent studies have put forward the advantages of using l-MBOT in SMT [Maletti, 2011] but a real system has never been implemented. As a consequence, the applicability of l-MBOT to SMT has never been empirically evaluated. A first contribution of this thesis closes this gap and builds a SMT system on a shallow variant of l-MBOT. Our work also provides an empirical evaluation of this system.

In this chapter, we begin by presenting several synchronous grammar models in increasing order of expressivity[1] (Section 2.1). The main goal of this presentation is to introduce l-MBOT, which is the formalism that we implement, and the baseline models we will compare to. In Section 2.2, we present previous work on building syntax-based SMT systems using the grammar formalisms presented before. We close this chapter by presenting our contributions to research on hard syntactic constraints for statistical machine translation.

2.1 Synchronous Grammars

2.1.1 Weighted Synchronous Context-Free Grammars

Weighted Synchronous Context-Free Grammars (SCFG) have been studied in [Aho and Ullman, 1969]. We first introduce weighted CFG [Hopcroft et al., 2006] and then extend them to weighted SCFG.

Weighted Context-Free Grammars

Formal Definition A Context-Free Grammar (CFG) is a grammar $G = (N, \Sigma, P, S)$ where N is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols, $S \in N$ the start non-terminal and P a finite set of grammar rules. Each CFG rule has the form $A \rightarrow \alpha$, with $\alpha \in (N \cup \Sigma)^*$ and $A \in N$. A is the *left-hand side* (lhs) of the rule and α its *right-hand-side* (rhs). The semantics of the CFG is given by the following rewrite relation: if $A \rightarrow \alpha \in P$ and $\beta \in (N \cup \Sigma)^*$ and $\gamma \in (N \cup \Sigma)^*$, then $\beta A \gamma \underset{G}{\Rightarrow} \beta \alpha \gamma$. In other words, if the lhs of a rule appears in a string then it can be replaced by the rhs of the rule. The rewriting of the start symbol S into a string t of terminal symbols is called a derivation for t. The language of G is the set of all strings t that have a derivation. Formally, $L(G) = \{t \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} t\}$.
A weighted CFG is a grammar $G = (N, \Sigma, P, S, w)$ where (N, Σ, P, S) is a CFG and

[1] Another overview is given in [Chiang, 2006].

$w : P \to \mathbb{R}$ is a function that assigns a weight to each grammar rule. A probabilistic CFG is a weighted CFG where $w : P \to [0, 1]$ and the weights of all rules with the same lhs sum to 1. The weight of a derivation is the product of the weights of the rules used in the derivation. In the remainder of this chapter, the rule weights are indicated over the arrow (see rule r_1 below).

Example We present a weighted CFG $G = (N, \Sigma, P, S, w)$ for a tiny portion of French. The set of non-terminals is $N = \{$*NP*, *DET*, *NN*$\}$. The set of terminals is $\Sigma = \{$*une*, *approche*, *pratique*$\}$. The start symbol S is *NP*. The set P of rules consists of r_1 to r_4 :

r_1 *NP* $\xrightarrow{1.0}$ *DET NN*

r_2 *DET* $\xrightarrow{1.0}$ *une*

r_3 *NN* $\xrightarrow{0.5}$ *approche*

r_4 *NN* $\xrightarrow{0.5}$ *pratique*

Rule r_1 can be used to rewrite the start symbol *NP* because this symbol appears in its lhs. The application of r_1 to *NP* yields the string *DET NN* which is the rhs of r_1. This derivation step is the first step in the derivation D below. To simplify the presentation, we indicate the rule instead of the grammar below the derivation arrow. In the same fashion, rule r_2 can be used to rewrite the string *DET NN* because the lhs *DET* of r_2 appears in this string. After rewriting, the string *une NN* is created. Following the same mechanism, the application of rule r_3 yields *une approche*. The weight of D is 0.5, that is the multiplication of the weights of r_1, r_2 and r_3.

D *NP* $\underset{r_1}{\Rightarrow}$ *DET NN* $\underset{r_2}{\Rightarrow}$ *une NN* $\underset{r_3}{\Rightarrow}$ *une approche*

The weighted CFG G recognizes the noun phrases *une approche* and *une pratique*. The weights of the derivations used in this process are 0.5 each.

Weighted Synchronous Context-Free Grammars

Formal Definition A Synchronous Context-Free Grammar (SCFG) is essentially a combination of 2 CFG $G_s = (N_s, \Sigma, P, S_s)$ and $G_t = (N_t, \Delta, P, S_t)$. Formally, it is a system[2] $G = (N_s, N_t, \Sigma, \Delta, P, S_s, S_t)$ where P is a finite set of grammar rules (or productions) of the form $(A_s, A_t) \rightarrow \langle \alpha, \beta, \widetilde{A} \rangle$ such that $A_s \rightarrow \alpha \in P_s$, $A_t \rightarrow \beta \in P_t$ and the number of nonterminal occurrences in α and β coincide. The alignment function $\widetilde{A}$ is a one-to-one correspondence between non-terminals in α and β such that the i-th input non-terminal (read from left to right) in α is the same as the $\widetilde{A}(i)$-th output non-terminal in β. If α has k non-terminal occurrences, then $\widetilde{A} : \{1, \cdots, k\} \rightarrow \{1, \cdots, k\}$ is a permutation from $\{1, \cdots, k\}$ to $\{1, \cdots, k\}$. We write $\widetilde{A}$ as $[\widetilde{A}(1), \widetilde{A}(2), ..., \widetilde{A}(k)]$. The semantics of SCFG is given by the following rewrite relation on sentential forms, which have exactly the same shape as our right-hand sides of productions: Given a sentential form $(\alpha_1, \beta_1, \widetilde{A_1})$ and a production $(A_s, A_t) \rightarrow (\gamma, \delta, \widetilde{A})$ such that:

1. A_s is the i-th non-terminal in α_1
2. A_t is the $\widetilde{A_1}(i)$-th non-terminal in β_1,

$(\alpha_1, \beta_1, \widetilde{A_1})$ can be rewritten into $(\alpha_2, \beta_2, \widetilde{A_2})$ where α_2 is obtained by replacing the i-th non-terminal A_s in α_1 by γ. In the same fashion, β_2 is obtained by replacing the $\widetilde{A_1}(i)$-th non-terminal A_t in β_1 by δ. Finally, $\widetilde{A_2}$ is defined below, where m is the number of non-terminals in γ.

$$\forall d \mid 1 \leq d \leq m, \widetilde{A_2}(i + d - 1) = \widetilde{A_1}(i) + \widetilde{A}(d) - 1 \tag{2.1}$$

$$\forall j > i, \widetilde{A_2}(j + m - 1) = \begin{cases} \widetilde{A_1}(j) + m - 1 & \text{if } \widetilde{A_1}(j) > \widetilde{A_1}(i) \\ \widetilde{A_1}(j) & \text{otherwise} \end{cases} \tag{2.2}$$

[2]Note that the definition of syntax directed translation scheme given in [Aho and Ullman, 1969] only augments the considered CFG with a set of output terminal symbols. This definition is too restrictive for an application to syntax-based machine translation with linguistic annotations, where the parse labels vary from one language to the other.

$$\forall j > i,\ \widetilde{A}_2(j) = \begin{cases} \widetilde{A}_1(j) + m - 1 & \text{if } \widetilde{A}_1(j) > \widetilde{A}_1(i) \\ \widetilde{A}_1(j) & \text{otherwise} \end{cases} \tag{2.3}$$

In other words, all non-terminals behind $\widetilde{A}_1(i)$ are moved $(m-1)$ places further. For every integer k let $\langle k \rangle$ be k if $k < \widetilde{A}_1(i)$ and $k + (m-1)$ otherwise. Then $\widetilde{A}_2 = [\langle \widetilde{A}_1(1) \rangle, ..., \langle \widetilde{A}(i-1) \rangle, \widetilde{A}_1(i) + \widetilde{A}(1) - 1, ..., \widetilde{A}_1(i) + \widetilde{A}(m) - 1, \langle \widetilde{A}_1(i+1) \rangle, ..., \langle \widetilde{A}_1(n) \rangle]$. The rewrite relation for SCFG is written as $(\alpha_1, \beta_1, \widetilde{A}_1) \underset{G}{\Rightarrow} (\alpha_2, \beta_2, \widetilde{A}_2)$. Whenever G is obvious from the context we might instead annotate the rule. The synchronous context-free languages are exactly the sets of string pairs generated by SCFG. More precisely, the language generated by G is the set of pairs of terminal strings obtained by recursively rewriting the start symbols (S_s, S_t) using rules of G. Formally, $L(G) = \{(w_s, w_t) \in \Sigma^* \times \Delta^* \mid (S_s, S_t, [1]) \overset{*}{\underset{G}{\Rightarrow}} (w_s, w_t, [])\}$. The rewriting of the start symbols (S_s, S_t) into a pair of strings of terminal symbols is called a derivation.
A weighted SCFG is defined in the same way as a weighted CFG, by assigning a weight to each synchronous grammar rule. As in CFG, the weight of a derivation is the product of the weights of the rules used in this derivation.

Example We present a weighted (and probabilistic) SCFG $G = (N_s, N_t, \Sigma, \Delta, P, S_s, S_t, w)$ for a tiny portion of French and English. The set of input non-terminals is $N_s = \{$*S*, *NP*, *ADJ*, *NN*, *PP*$\}$. The set of output non-terminals is $N_t = \{$*TOP*, *NP*, *ADJ*, *NN*, *PP*$\}$. The set of input terminals is $\Sigma = \{$*une*, *étude*, *approche*, *pratique*, *du*, *document* $\}$ and the set of output terminals is $\Delta = \{$*a*, *study*, *approach*, *practical, of, the, document*$\}$. The start symbol (S_s, S_t) is *(S,TOP)*. The set P of rules contains r_1 to r_6:

r_1 *(S,TOP)* $\xrightarrow{1.0}$ $\langle$ *NP PP*, *NP PP*, $[1, 2]$ $\rangle$

r_2 *(NP,NP)* $\xrightarrow{1.0}$ $\langle$ *une NN ADJ*, *a ADJ NN*, $[2, 1]$ $\rangle$

r_3 *(ADJ,ADJ)* $\xrightarrow{1.0}$ $\langle$ *pratique*, *practical* $\rangle$

r_4 *(NN,NN)* $\xrightarrow{0.5}$ ⟨ *étude, study* ⟩

r_5 *(NN,NN)* $\xrightarrow{0.5}$ ⟨ *approche, approach* ⟩

The alignment $\widetilde{A}$ of each rule indicates the correspondence between non-terminal symbols. For instance, in rule r_1, we have $\widetilde{A}(1) = 1$, which means that the first non-terminal in the lhs (*NP*) corresponds to the first non-terminal in the rhs (*NP*). In the same fashion, $\widetilde{A}(2) = 2$ indicates that the second non-terminal in the lhs (*PP*) corresponds to the second non-terminal in the rhs (*PP*). In rule r_2, we have $\widetilde{A}'(1) = 2$, which means that the first non-terminal in the lhs (*NN*) corresponds to the second non-terminal in the rhs (*NN*). Note that when there are no non-terminals in the right-hand sides, such as in r_3, the permutation is omitted from the rule.

Rule r_1 can be used to rewrite the sentential form *(S , TOP,[1])* because this non-terminal pair appears in the lhs of r_1. This derivation step, denoted by D_1 below, yields the sentential form *(NP PP , NP PP)* with alignment $\widetilde{A} = [1, 2]$. This string pair can be rewritten using r_2 because the non-terminal *NP* in the lhs of r_2 appears (i) at position 1 in the input side and (ii) at position $\widetilde{A}(1) = 1$ in the output side of the considered string. This second derivation step, denoted by D_2 below, yields the sentential form *(une NN ADJ PP, a ADJ NN PP,* $\widetilde{A}'$*)* with alignment $\widetilde{A}' = [2, 1, 3]$. This string pair is created in three steps, illustrated below. First, the input side is obtained by replacing the non-terminal *NP* (in box) by the string *une NN ADJ* which is the input rhs of r_2. In a second step, the output side is obtained by replacing the non-terminal *NP* by the string *a ADJ NN*, which is the output rhs of r_2.

D_1 ⟨ *S,TOP,[1]* ⟩ $\underset{r_1}{\Rightarrow}$ ⟨ *NP PP, NP PP,* $[1, 2]$ ⟩

D_2 ⟨ $\boxed{\textit{NP}}$ *PP,* $\boxed{\textit{NP}}$ *PP,* $[1, 2]$ ⟩ $\underset{r_2}{\Rightarrow}$ ⟨ $\boxed{\textit{une NN ADJ}}$ *PP,* $\boxed{\textit{a ADJ NN}}$ *PP,* $[2, 1, 3]$ ⟩

Finally, the alignment is created as illustrated below: there are three non-terminal pairs in the sentential form obtained after step D_2, so $\widetilde{A}'$ has length 3.

Because the substituted non-terminal *NP* (in box) is at position 1 in the input string pair, the alignment of r_2 fills the two first slots of $\widetilde{A}'$, following Equation 2.1 above.

D_2 $\langle$ $\boxed{NP}$ *PP*, $\boxed{NP}$ *PP*, $[1, 2]$ $\rangle \underset{r_2}{\Rightarrow}$ $\langle$ une ***NN ADJ*** *PP*, *a* ***ADJ NN*** *PP*, $[\mathbf{2}, \mathbf{1}, 3]$ $\rangle$

r_2 (*NP*,*NP*) $\xrightarrow{1.0}$ $\langle$ *une* $\boxed{NN\ ADJ}$, a $\boxed{ADJ\ NN}$, **[2,1]** $\rangle$

As the non-terminal *PP* (in box below) follows *NP* in the input string, its alignment occupies the last slot in $\widetilde{A}'$. Because *PP* is preceded by two non-terminals, its alignment is increased by $2 - 1 = 1$, following Equation 2.2 above.

D_2 $\langle$ *NP* $\boxed{PP}$, *NP* $\boxed{PP}$, $[1, 2]$ $\rangle \underset{r_2}{\Rightarrow}$ $\langle$ *une NN ADJ* ***PP***, *a ADJ NN* ***PP***, $[2, 1, \mathbf{3}]$ $\rangle$

Following the same mechanism, rule r_3 can be used to rewrite the sentential form obtained after step D_2 by replacing the non-terminal *ADJ* by the input and output rhs of r_3.

D_3 $\langle$ *une NN* $\boxed{ADJ}$ *PP*, a $\boxed{ADJ}$ *NN PP*, $[2, 1, 3]$ $\rangle$
$\underset{r_2}{\Rightarrow}$ $\langle$ *une NN* $\boxed{pratique}$ *PP*, *a* $\boxed{practical}$ *NN PP*, $[1, 2]$ $\rangle$

The synchronous context-free language generated by G is the pair of noun phrases *une étude pratique du document, a practical study of the document* and *une approche pratique du document, a practical approach of the document*.

2.1.2 Weighted Synchronous Tree Substitution Grammars

SCFG generate a language that consists of pairs of strings. When applied to trees, such grammars can only perform transformations of depth one, i.e. only work on sister nodes. Synchronous Tree Substitution Grammars (STSG) [Eisner, 2003] allow more complex transformations by modeling pairs of trees instead of pairs of strings. Unlike SCFG, STSG can reorder nodes that are at different levels in a tree. For instance it can swap the tree fragments rooted at NP and PP when transforming the French parse tree in Figure 2.8 into its English counterpart.

Weighted Tree Substitution Grammars

Formal Definition A Tree Substitution Grammar (TSG) is a grammar $G = (N, \Sigma, P, S)$ where N is a set of non-terminal symbols, Σ is a set of terminal symbols, S the start non-terminal and P the finite set of grammar rules. Each TSG rule is called an elementary tree and has the form $T = (V, V_i, E, m, l, r)$ where V is an ordered finite set of nodes, $V_i \subseteq V$ a set of internal nodes and $E \subseteq V \times V$ a set of directed edges such that (V, E) is a rooted connected tree. The set $V_l = V - V_i$ is the set of leaf nodes. The function $m : V_i \rightarrow N$ labels each internal node with a non-terminal symbol. The function $l : V_l \rightarrow (N \cup \Sigma)$ labels each leaf node with a label from $(N \cup \Sigma)$. The symbol r is the label of the root node of a tree. Following [Eisner, 2003], we use the notation $T.$ to access the elements of a tree T. For instance, we write $T.V$ to refer to the set of nodes of T. The semantics of TSG is given by the following rewrite relation: let T_1 and T_2 be elementary trees and $d \in T_1.V_l$ a leaf node of T_1. If $T_1.l(d) = T_2.r$ then we can use T_2 to rewrite T_1 into T_{12} by substituting d in T_1 by T_2. Let $v \in T_1.V$ be the node such that $(v, d) \in T_1.E$ provided that such a node exists and let $w \in T_2.V$ be the root of T_2. The substitution operation yields $T_{12} = (V_{12}, V_{i12}, E_{12}, m_{12}, l_{12}, r_{12})$ where:[3]

$$V_{12} = (T_1.V - \{d\}) \cup T_2.V$$

$$V_{i12} = T_1.V_i \cup T_2.V_i$$

$$E_{12} = \begin{cases} T_1.E \cup T_2.E - \{(v,d)\} \cup \{(v,w)\} & \text{if } d \text{ is not the root} \\ T_2.E & \text{otherwise} \end{cases}$$

$$m_{12}(q) = \begin{cases} T_1.m(q) & \text{if } q \in T_1.V_i \\ T_2.m(q) & \text{if } q \in T_2.V_i \end{cases}$$

$$r_{12} = \begin{cases} T_1.r & \text{if } d \text{ is not the root of } T_1 \\ T_2.r & \text{otherwise} \end{cases}$$

[3] We assume that $T_1.V$ and $T_2.V$ are disjoint.

$$l_{12} = (T_1.l \cup T_2.l) - \{(d, T_2.r)\}$$

The rewrite relation for TSG can be written as $T \underset{G}{\Rightarrow} T'$. Each set of trees generated by a TSG is a tree substitution language. More precisely, the generated language is the set of trees obtained by recursively rewriting trees with root label S until all leaf nodes with labels in N are substituted. Formally, $L(G) = \{t \mid T_s \in P \text{ is elementary and } T_s \underset{G}{\overset{*}{\Rightarrow}} t \text{ with } T_s.r = S \text{ and } t.l(t.V_l) \cap N = \emptyset\}$. The rewriting of an elementary tree T into t using the grammar G is called a derivation.
A weighted TSG is a grammar $G = (N, \Sigma, P, S, w)$ where $w : P \to \mathbb{R}$ is a function that assigns a weight to each elementary tree. A probabilistic TSG is a weighted TSG where $w : P \to [0, 1]$ and the weight of all trees with the same root label sums to 1. The weight of a derivation is the product of the weights of the elementary trees used in this derivation.

Example We present an example weighted TSG $G = (N, \Sigma, P, S, w)$ for a tiny portion of French. The set of internal node labels is $N = \{S, NP, DT, ADJ, NN, VP, V\}$ and the set of leaf node labels is $N \cup \Sigma$ with $\Sigma = \{this, report, lacks, a, practical, study\}$. The start symbol is S. The set P of rules consists of T_1 to T_5 shown in Figures 2.1 and 2.2. The weight of each tree is given in the caption.

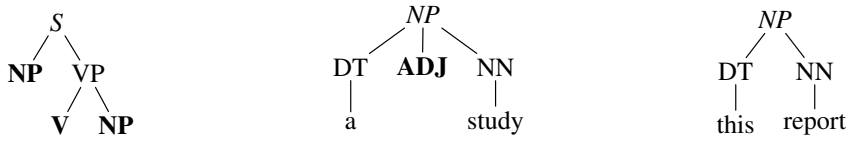


Figure 2.1: Elementary trees T_1 to T_3 with weights 1.0, 0.5 and 0.5

Figure 2.2: Elementary trees T_4 and T_5 with (both) weights 1.0

We assume that the tree in Figure 2.1 is the starting point of the derivation process as its root label is the start symbol S. Rule T_2 can be used to rewrite T_1 because its root label (NP) is equal to a leaf label of T_1. This derivation step can yield the tree T_{12} in Figure 2.1.2, where T_2 substitutes the leftmost leaf label NP in T_1. Note that all leaves labeled with NP could be replaced by T_2. In the same fashion, rule T_5 can be used to rewrite tree T_{12} because its root label (ADJ) is equal to the label of a non-terminal leaf of T_{12}. The created tree T_{125} is obtained by substituting the node labeled with ADJ in T_{12} by T_5.

Figure 2.3: Elementary Trees T_{12} and T_{125}

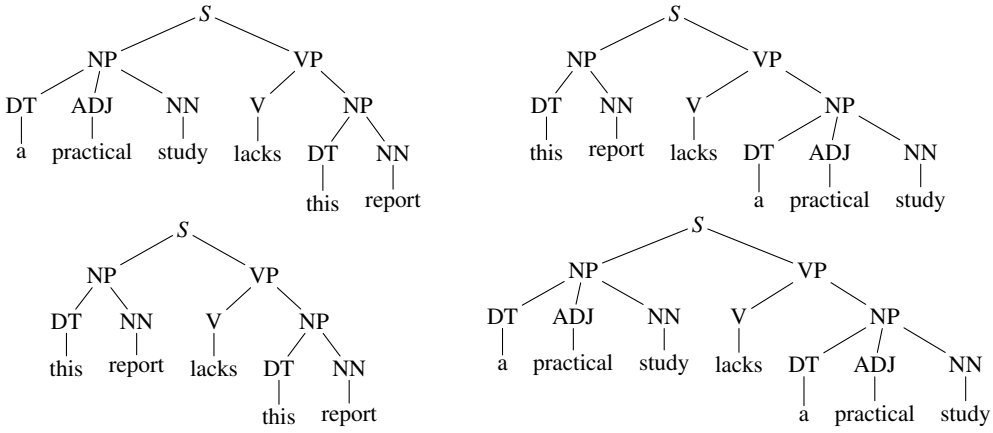


Figure 2.4: Trees in the language generated by TSG G

The Tree Substitution Language generated by G consists of the four trees given in Figure 2.4. Their derivations have weight 0.25 each.

Weighted Synchronous Tree Substitution Grammars

Formal Definition A Synchronous Tree Substitution Grammar (STSG) is essentially a combination of two TSG $G_s = (N_s, \Sigma, P_s, S_s)$ and $G_t = (N_t, \Sigma, P_t, S_t)$. Formally, it is a system $G = (N_s, N_t, \Sigma, \Delta, P, S_s, S_t)$. The rules P are elementary tree pairs $T_p = (T_1, T_2, \widetilde{A})$ where $T_1 \in P_s$, $T_2 \in P_t$, and T_1 contains as many non-terminal leaves as T_2. We write $T_1.nt$ and $T_2.nt$ to denote the sets of leaf nodes in $T_1.V_l$ and $T_2.V_l$ that are labeled by non-terminals. The alignment is a bijective function $\widetilde{A} : T_1.nt \rightarrow T_2.nt$. The semantics of STSG is given by the following rewrite relation. Assume that T_{p1} and T_{p2} are elementary tree pairs. If conditions (i) and (ii) below are met, then we can use T_{p2} to rewrite T_{p1} into T_{p12}:

(i) a leaf node $n \in T_{p1}.T_1.nt$ is labeled by the root label $T_{p2}.T_1.r$

(ii) the aligned leaf node $\widetilde{A}(n) \in T_{p1}.T_2.nt$ is labeled by the root label $T_{p2}.T_2.r$

The rewriting is done by substituting n with $T_{p2}.T_1$ and $\widetilde{A}(n)$ with $T_{p2}.T_2$ according to the substitution procedure defined in Section 2.1.2. The rewrite relation for STSG can be written as $T_p \underset{G}{\Rightarrow} T_{ps}$. Each set of tree pairs generated by a STSG is a Synchronous Tree Substitution Language. More precisely, the generated language is the set of tree pairs obtained by recursively rewriting a chosen start tree pair T_{ps} until all leaf nodes labeled with non-terminals are substituted. Formally, $L(G) = \{T_p \mid t_p$ is a tree pair $\wedge\, T_{ps} \overset{*}{\underset{G}{\Rightarrow}} t_p$ with $T_{ps}.T_1.r = S_s$ and $T_{ps}.T_2.r = S_t$ and $t_p.T_1.nt = t_p.T_2.nt = \emptyset\}$. The rewriting of an elementary tree pair T_{p1} into T_{p12} is called a derivation.

A weighted STSG is defined in the same way as a weighted TSG, by assigning a weight to each tree pair. As in TSG, the weight of a derivation is the product of the weights of the tree pairs used in this derivation.

Example We present an example weighted STSG $G = (N_s, N_t, \Sigma, \Delta, P, S_s, S_t, w)$ for a small portion of French and English. The elementary tree pairs are given in

Figures 2.5 to 2.6. The start symbols are $S_s = TOP$ and $S_t = S$. The set of source and target non-terminals are $N_s = \{TOP, NP, VP, V, PP, DT, NN, ADJ, PREP\}$ and $N_t = \{S, NP, VP, V, DT, ADJ, NN\}$. The set of source and target terminal symbols are $\Sigma = \{une, étude, pratique, manque, à, ce, rapport\}$ and $\Delta = \{a, study, practical, lacks, this, report\}$. The alignment function is indicated by blue links. We assume that all tree pairs have weight 1.0.

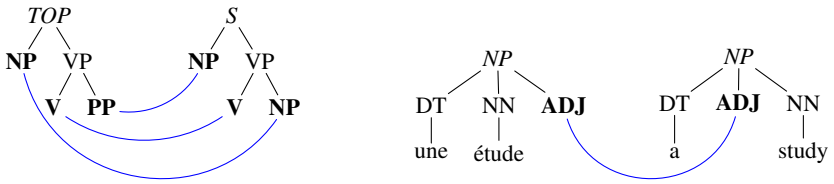


Figure 2.5: Tree pairs T_{p1} and T_{p2}

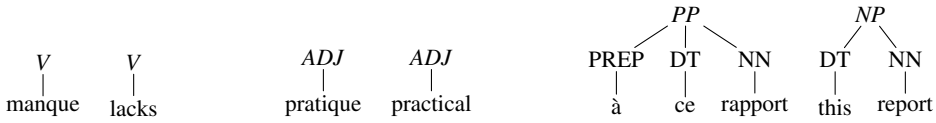


Figure 2.6: Tree pairs T_{p3} to T_{p5}

We assume that the tree pair T_{p1} in Figure 2.5 is the starting point of the derivation because the root labels of its elementary trees are the start symbols TOP and S. Rule T_{p2} can be used to rewrite T_{p1} because:

(i) a leaf node n in $T_{p1}.T_1.nt$ is labeled by the root label $T_{p2}.T_1.r$ (NP)

(ii) the leaf node in $T_{p1}.\widetilde{A}(w)$ is labeled by the root label $T_{p2}.T_2.r$ (NP)

During this derivation step, the chosen leaves labeled with NP in T_{p1} are replaced by $T_{p2}.T_1$ and $T_{p2}.T_2$, yielding the tree pair T_{p12} in Figure 2.7. This tree pair can be further processed using rule T_{p4} yielding the tree pair T_{p25} illustrated in Figure 2.7.

The synchronous tree substitution language generated by G is the tree pair given in Figure 2.8. The weight of its derivation is 1.0.

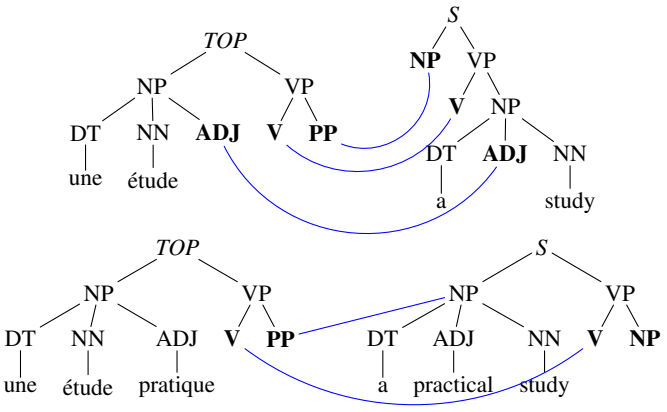


Figure 2.7: Tree pairs T_{p12} and T_{p25}

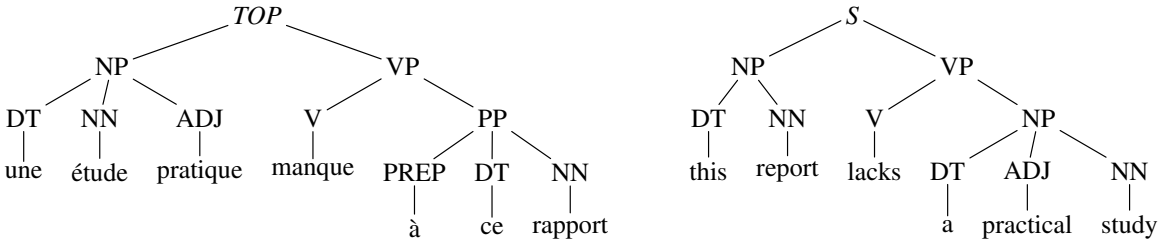


Figure 2.8: Tree pair generated by STSG G

2.1.3 Weighted Synchronous Tree Sequence Substitution Grammars

By modeling relations between trees, STSG can perform tree transformations such as subtree swapping at different levels. However these grammars only allow to generate complete tree fragments such as the subtrees rooted in NP and PP in Figure 2.8. But many applications require to perform transformations between segments that do not span a complete tree fragment. For instance, in Figure 2.9, the string *ne manque pas*, which translates to *does not lack*, does not span a complete tree. Hence, a formalism that only performs transformations on trees, such as STSG, forces a system to include the sentence structure of the French tree in Figure 2.9 into each rule for "ne manque pas". Synchronous Tree Sequence Substitution Grammars (STSSG) model relations between sequences of trees instead of single trees. Such grammars allow to transform tree sequences into each other which is very useful when working with highly non-isomorphic trees such as the tree pair in Figure 2.9. In this case, a STSSG can transform the tree sequence that spans the string *ne manque pas* (2 trees) into the tree sequence that spans *does not lack* (3 trees). The STSSG rule performing this transformation is given in Figure 2.11.

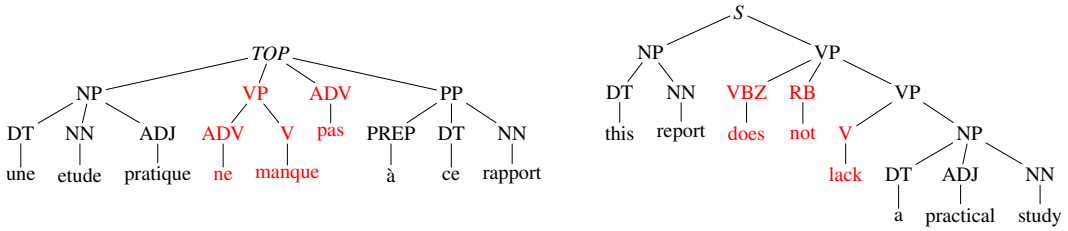


Figure 2.9: Non-isomorphic trees. Contiguous tree sequences are shown in red.

Two versions of STSSG have been suggested which propose a different definition of grammar rules and alignment. [Zhang et al., 2008] present a version of STSSG where the tree sequences in the grammar rules have contiguous spans such as the sequences presented above. This type of grammar handles well cases of non-isomorphic trees where contiguous spans correspond to each other. In many cases,

however, tree sequences that translate into each other are not contiguous. This phenomenon occurs frequently in language pairs such as English and German, Arabic or Chinese. An example tree pair with non-contiguous tree sequences is given in Figure 2.10. In order to model that kind of transformation, [Sun et al., 2009] define a non-contiguous version of STSSG.

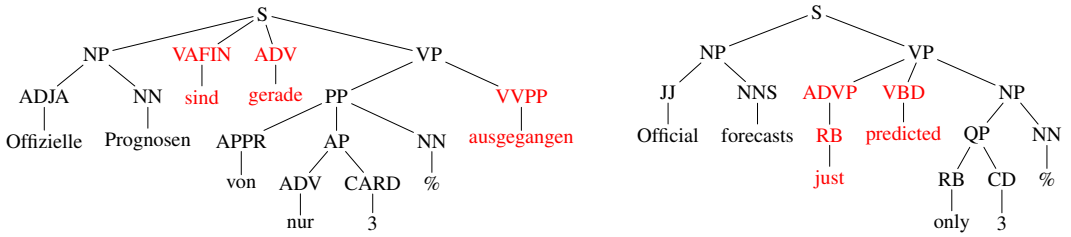


Figure 2.10: Non-isomorphic trees. Non-contiguous tree sequences are shown in red.

Formal Definition

A STSSG is essentially a STSG where grammar rules consist of sequences of elementary trees on the input and output side instead of single elementary trees. Formally an STSSG is a grammar $G = (N_s, N_t, \Sigma, \Delta, P, S_s, S_t)$ where N_s, N_t, Σ, Δ, S_s and S_t are as for STSG. P is a finite set of productions explained next.

Contiguous Synchronous Tree Sequence Substitution Grammars

In a contiguous STSSG, the grammar rules are tree sequence pairs $T_{ps} = (T_{s1}, T_{s2}, \widetilde{A})$ where T_{s1} and T_{s2} are sequences of (input and output) elementary trees as defined in Section 2.1.2. We write $T_{s1}.nt$ and $T_{s2}.nt$ to denote the sequence of leaves (taken left-to-right) labeled with non-terminals in the trees (taken left-to-right) composing T_{s1} and T_{s2}. The alignment $\widetilde{A} \subseteq T_{s1}.nt \times T_{s2}.nt$ is a relation from the set of non-terminal leaves $T_{s1}.nt$ to the non-terminal leaves $T_{s2}.nt$. The semantics of STSSG is given by the following rewrite relation. Suppose that T_{ps1} and T_{ps2} are tree sequence pairs with alignment $\widetilde{A}$. If conditions (i) and (ii) below are met, then we can use

T_{ps2} to rewrite T_{ps1} into T_{ps12}. We write $T_{s1}.r$ to denote the ordered set of roots of the trees (taken left-to-right) in the sequence T_{s1}.

(i) there exists an (ordered) set of contiguous leaf nodes $\{l_1, \cdots, l_m\} \subseteq T_{ps1}.T_{s1}.nt$ labeled by the sequence of roots $T_{ps2}.T_{s1}.r$

(ii) the (ordered) set of contiguous aligned leaf nodes $\widetilde{A}(\{l_1, \cdots, l_m\}) \subseteq T_{ps1}.T_{s2}.nt$ is labeled by the sequence of roots $T_{ps2}.T_{s2}.r$

The rewriting is done by substituting $\{l_1, \cdots, l_m\}$ by $T_{ps2}.T_{s1}$ and $\widetilde{A}(\{l_1, \cdots, l_m\})$ by $T_{ps2}.T_{s2}$ according to the substitution procedure defined in Section 2.1.2. The rewrite relation for STSSG can be written as $T_{ps} \underset{G}{\Rightarrow} t_{ps}$. Each set of pairs of trees generated by a STSSG is a synchronous tree sequence substitution language. The generated language is written in exactly the same way as for STSG except that the grammar rules used in a derivation are from the STSSG, i.e. tree sequence pairs. A weighted STSSG is defined in the same way as a weighted STSG, by assigning a weight to each pair of tree sequences. The weight of a derivation is the product of the weights of rules used in this derivation.

Example We present an example weighted contiguous STSSG G for a tiny portion of French and English. Our grammar consists of the elementary tree pairs presented in Figures 2.5 and 2.6 (in Section 2.1.2) and Figure 2.12 below. Note that elementary tree pairs are a particular case of tree sequence pairs, where the length of the sequences is 1 and the alignment between non-terminals is a one-to-one correspondence. The rules of G also contain the tree sequence pairs given in Figures 2.11 and 2.12. The one-to-one alignments between pairs of leaves is indicated by blue links. The many-to-many alignments are indicated in red. We assume that (i) all tree pairs have weight 1.0 and (ii) the tree sequence pair T_{ps6} is the starting point of the derivation because the root labels of its trees are the start symbols. This tree pair can be rewritten using the tree sequence pair T_{ps1} because conditions (i) and (ii) below are met:

(i) a set of leaf nodes $T_{ps6}.T_{s1}.nt$ is labeled by the sequence of root labels in $T_{ps1}.T_{s1}.r$ (VP, ADV)

(ii) the set of leaf nodes $\widetilde{A}(VP, ADV)$ is labelled by the sequence of roots in $T_{ps1}.T_{s2}.r$ (VBZ, RB, V)

The created tree pair T_{p61} is obtained by substituting the nodes labeled $\{VP, ADV\}$ in $T_{ps6}.T_1.nt$ by $T_{ps1}.T_{s1}$ and the nodes labeled $\{VBZ, RB, V\}$ in $T_{ps6}.T_2.nt$ with $T_{ps1}.T_{s2}$.

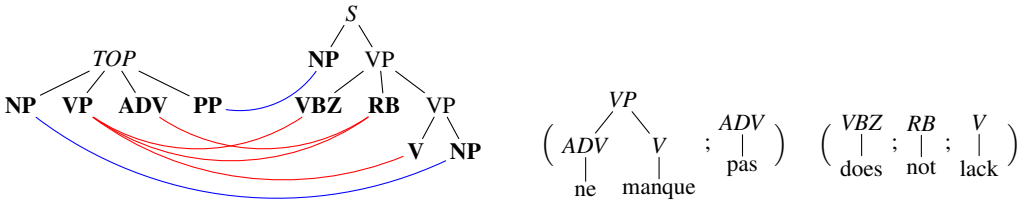


Figure 2.11: Tree sequence pairs T_{ps6} and T_{ps1}

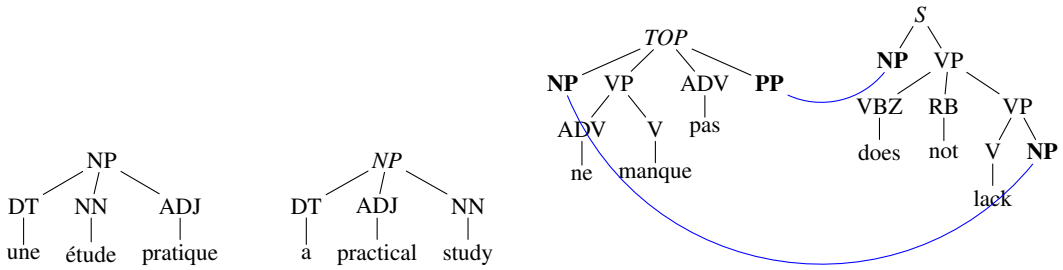


Figure 2.12: Tree pairs T_{p7} and T_{p61}

By substituting the leaves labeled NP and PP in $T_{p61}.T_1.nt$ and the aligned leaves NP and NP in $T_{p61}.T_2.nt$ by the tree pairs T_{p5} and T_{p7}, we obtain the tree pair shown in Figure 2.9. As no leaf nodes can be further substituted in this tree pair, it belongs to the language generated by our example contiguous STSSG. This tree is the only element of the language generated by our grammar. The weight of its derivation is 1.0.

Non-contiguous Synchronous Tree Sequence Substitution Grammars

A non-contiguous Synchronous Tree Sequence Substitution Grammar (Nc-STSSG) is essentially a contiguous STSSG without the restriction that aligned leaves labeled with non-terminals have to be contiguous. We present an example weighted Nc-STSSG G for a tiny portion of German and English. Our grammar consists of the tree sequences presented in Figures 2.13 to 2.15. The display of many-to-many alignments as well as labeled root states is the same as for contiguous STSSG (Section 2.1.3). The rewriting process works exactly like for contiguous STSSG except for contiguity. For instance, the non-contiguous tree sequence T_{ncp4} can be used to rewrite T_{ncp1} (which we assume to be the starting point of the derivation process) into T_{ncp41} shown in Figure 2.16. Further rewriting of T_{ncp41} using T_{ncp2} and T_{ncp3} yields the tree pair shown in Figure 2.10. As this pair is the only pair that can be generated by our example grammar G, it constitutes the language of G.

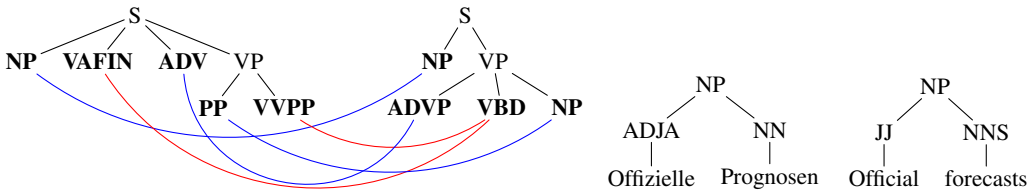


Figure 2.13: Tree pairs T_{ncp1} and T_{ncp2}

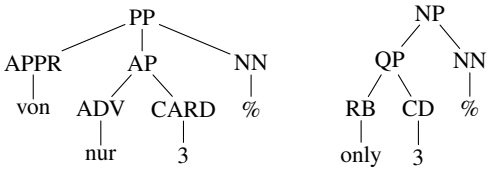


Figure 2.14: Tree pair T_{ncp3}.

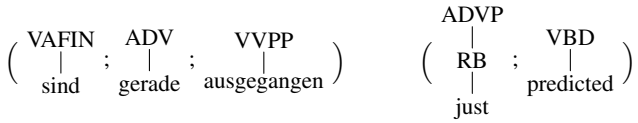


Figure 2.15: Tree sequence pair T_{ncp4}.

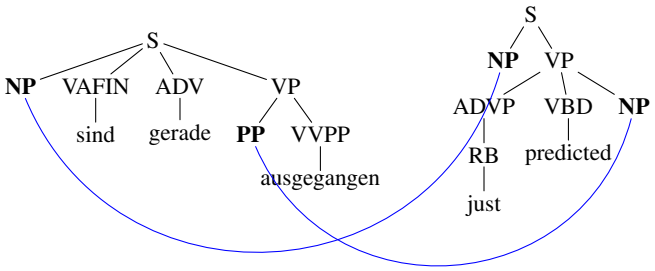


Figure 2.16: Tree pair T_{ncp41}.

Weighted Local Multi Bottom Up Tree Transducers

While Nc-STSSG are well suited to capture long distance dependencies in highly non-isomorphic tree pairs, they may be too powerful for SMT applications. As we will show in Section 4, formal grammar rules for SMT are extracted from tree pairs with aligned leaves where the leaf alignments represent correspondences between the words of two languages. For instance, Figure 2.17 shows the tree pair from which the rules of the Nc-STSSG presented above (Section 2.1.3) could have been extracted. In this example, the German word *nur* is aligned to *only*.

Figure 2.18 shows the same sentence pair but where the German word *nur* is aligned to the English words *just* and *only*. When dealing with this alignment, using a Nc-STSSG would lead to the extraction of the rule shown in Figure 2.19. But this rule models a potentially incorrect translation which hurts translation quality. In [Sun et al., 2009] the authors note this problem and ignore non-contiguous translation rules having discontiguities on both the source and target language side. Instead of ignoring some rules in a too powerful grammar, [Maletti, 2011] proposes the local multi bottom up tree transducer (l-MBOT) which only allows rules

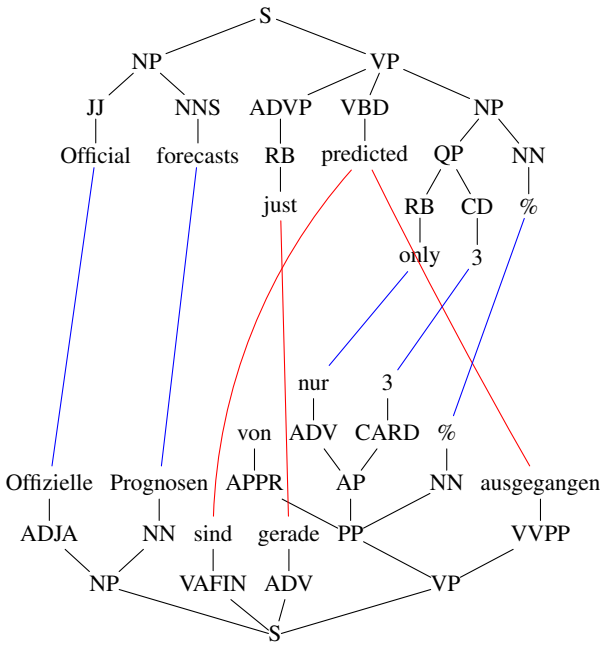


Figure 2.17: Non-isomorphic tree with aligned leaves representing a correct word alignment.

with tree sequences on the target language side. l-MBOT is even less powerful than non-contiguous STSSG ignoring discontiguities on source or target language side, as it forbids tree sequences on the input side even if those are contiguous while restricted Nc-STSSG allow contiguous tree sequences. By requiring a complete tree fragment on the input side of the rules, l-MBOT guarantees better syntactic coherence than non-contiguous STSSG. When working with noisy tree pairs such as shown in Figure 2.18, the grammar rules of l-MBOT would not contain the rule in Figure 2.19 as it is discontiguous on the input side.

Formal Definition l-MBOT is a Nc-STSSG with the restriction that the tree sequence in the input side of the rules has size 1, i.e. is an elementary tree as defined in Section 2.1.2. Formally, the grammar rules of l-MBOT are pairs $T_{ps} = (T_{p1}, T_{s2}, \widetilde{A})$ where T_{p1} is an elementary tree and T_{s2} is a sequence of trees. The alignment $\widetilde{A}$ is the same as in Nc-STSSG with the restriction that to each target there is exactly one

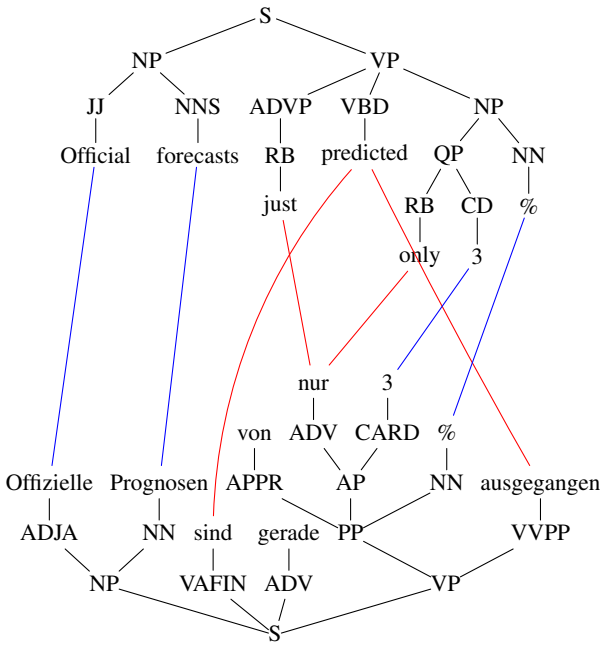


Figure 2.18: Non-isomorphic tree with aligned leaves.

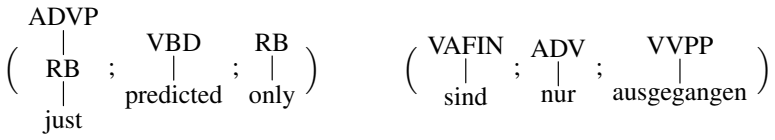


Figure 2.19: Tree sequence pair leading to a wrong translation.

source.

Example We present an example weighted l-MBOT G for a tiny portion of German and English. The rules of our grammar are shown in Figures 2.20 to 2.22. One-to-many alignments are displayed in red and root labels are given in italics. The rewriting process works in exactly the same way as for STSSG. As usual, we assume that all rules have weight 1.0 and that T_{ts1} is the starting point of our derivation. The l-MBOT rule T_{ts3} can be used to rewrite T_{ts1} into the tree pair T_{ts31} shown in Figure 2.22. Further rewriting of T_{ts31} using T_{ts2} and T_{ts4} yields the tree pair shown in Figure 2.23. As this pair is the only pair that can be generated by our example grammar G, it constitutes the language of G. The weight of its derivation is 1.0.

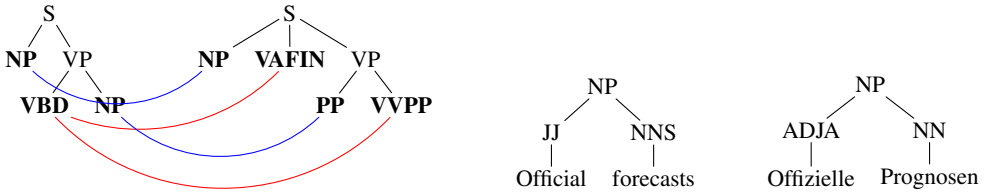


Figure 2.20: Tree pairs T_{ts1} and T_{ts2}

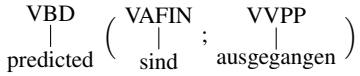


Figure 2.21: Tree-to-tree sequence pair T_{ts3}.

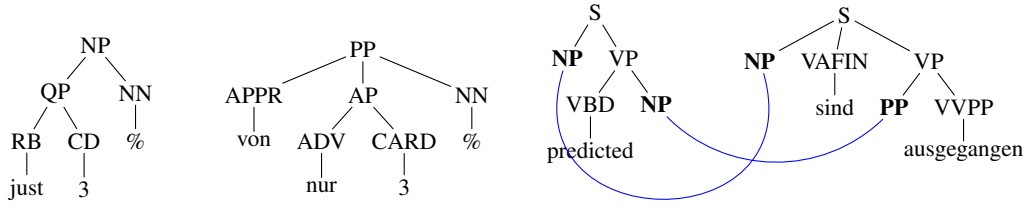


Figure 2.22: Tree pairs T_{ts4} and T_{ts31}

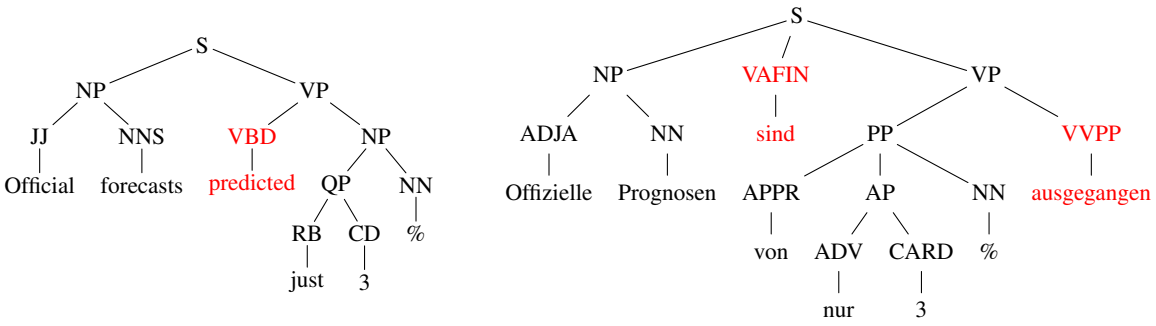


Figure 2.23: Tree pair generated by the l-MBOT grammar.

2.2 Statistical Machine Translation with Synchronous Grammars

Past work on syntax-based machine translation builds SMT systems based on the grammar formalisms presented above (Section 2.1). Among these, some only exploit the recursive structure of the grammar without including any linguistic annotation. In other systems, linguistic syntactic annotation is integrated into the grammar via source and target non-terminal alphabets. These usually consist of parse tree labels automatically acquired by parsing the source and target language side of the training data.[4] The integration of syntactic annotations is done at several levels: besides systems that use annotations on the source and target language side, some approaches integrate linguistic annotations of the source language only while others focus on target language annotations. We present previous work according to the formalism that is implemented and the amount of linguistic annotation that is integrated.

[4]We present examples of training data for such systems in Section 4.3

2.2.1 Weighted Synchronous Context-Free Grammars

A first line of work builds SMT systems using weighted Synchronous Context-Free Grammars (SCFG, presented in Section 4.3). Decoding for SCFG is usually done with CYK-style bottom-up chart parsing augmented with a translation generation component and language model scoring. We extensively present SCFG decoding in Sections 4.2 and 4.4.

Systems without Syntactic Annotations

Hierarchical SMT [Chiang, 2005, Chiang, 2007] is a SCFG-based system that integrates no syntactic annotation in its rules. In this framework, the input and output non-terminal alphabets are reduced to the single label X. Early work on this topic [Wu, 1997] presents a restricted version of non-annotated SCFG, called *inversion transduction grammars* (ITG). ITG rules cannot combine terminal and non-terminal symbols. Rules consisting of non-terminals have only two forms, which allow sequential translation or swapping. A middle ground between hierarchical grammars (HG) and ITG is the binary SCFG [Mylonakis and Sima'an, 2010]. This restricted form of SCFG (i) allows only 2 non-terminals on the source side of translation rules and (ii) only contains non-lexical or purely lexicalized rules. SMT systems based on binary SCFG [Mylonakis and Sima'an, 2010] yield translation quality comparable to hierarchical systems.

Many approaches have been proposed to improve the performance of hierarchical systems. An often highlighted weakness [Blunsom et al., 2008, Marton and Resnik, 2008, He et al., 2008] is the rule extraction and especially the scoring heuristic in [Chiang, 2005]. [Blunsom et al., 2008] built a discriminative translation model with derivations as latent variables which can perform rule scoring and decoding over multiple derivations. [Blunsom et al., 2009] use a non-parametric Bayesian model to induce grammar rules. None of these systems sig-

nificantly outperform hierarchical systems with the feature sets of [Chiang, 2005].[5] Further work improves hierarchical machine translation by defining syntactic features on hierarchical rules. As we contribute to this work, we dedicate an entire Chapter to a detailed presentation (in Chapter 3).

Hierarchical systems are implemented in many open source toolkits such as Moses [Hoang et al., 2009], Joshua [Li et al., 2009] or Jane [Vilar et al., 2012]. This allows the replication of the results in [Chiang, 2005] and the application of the approach to further language pairs. Our experiments (in Section 5.5) show that hierarchical systems are among the best performing syntax-based systems on some language pairs such as English-Arabic.

Systems with Source and Target Syntactic Annotations

Tree-to-tree SCFG-based SMT [Lavie et al., 2008, Ambati and Lavie, 2008, Lavie, 2008, Ambati et al., 2009] integrates syntactic annotation on the input and output alphabets of SCFG rules. As shown in [Ambati and Lavie, 2008, Ambati et al., 2009], tree-to-tree SCFG systems are too restrictive to achieve good translation quality: the syntactic structure of the input and output parse trees prevents such systems to extract translation rules with good lexical coverage. Moreover the shallow SCFG rules cannot model multi-level reorderings.[6] To avoid these shortcomings two strategies have been adopted. Several authors (see Section 2.2.2) use a more powerful formalism such as STSG to model tree-to-tree translation. Others (see Chapter 3) integrate source and target side syntactic information as features in a hierarchical model. Early work on tree-to-tree SCFG [Melamed, 2004] extends CFG parsing to induce SCFG and shows how to build an SMT system with the induced grammar.

Tree-to-tree SCFG are implemented in the Moses open source toolkit which

[5] In [Blunsom et al., 2008] language model scoring is not integrated in the decoding procedure, the system in [Blunsom et al., 2008] underperforms a hierarchical baseline trained with the features of [Chiang, 2005]

[6] [Zhang et al., 2007] show that this restriction notably hurts translation quality.

allows to perform SCFG based SMT on several language pairs. Our evaluation (in Section 5.5) shows that tree-to-tree SCFG systems achieve the worst performance among the syntax-based systems we tested.

Systems with target side syntactic annotations

String-to-tree SCFG machine translation [Zollmann and Venugopal, 2006, Almaghout et al., 2011, Williams and Koehn, 2012, Hanneman and Lavie, 2013] integrates syntactic annotation on the target side of SCFG rules. Among work on string-to-tree systems, [Zollmann and Venugopal, 2006] annotate the target side of SCFG rules with fuzzy labels from a tagset similar to Combinatory Categorial Grammars (CCG) [Steedman, 1996]. [Almaghout et al., 2011] simplify these labels by only considering the left and right context in the CCG annotations[7]. [Hanneman and Lavie, 2013] also reduce the set of CCG labels in [Zollmann and Venugopal, 2006] by clustering those using bilingual labels.[8] The authors report significant improvements over a hierarchical baseline. [Williams and Koehn, 2012] use the rule extraction procedure for string-to-tree STSG [Galley et al., 2004, Galley et al., 2006] with rule combination [DeNeefe et al., 2007] (see Section 2.2.2). They transform the obtained rules into a SCFG by removing the internal nodes in the target side trees.[9] The results of shared tasks such as [Bojar et al., 2014] show that the approach in [Williams and Koehn, 2012] ranks highly among other types of syntax-based and other SMT systems (e.g. phrase-based). Early work on string-to-tree decoding [Yamada and Knight, 2002] transforms strings into trees by learning a target language CFG and combining it with a set of rules that model the translation process. During decoding, a source tree is built on the input string which is then

[7] And thus removing the functor information

[8] More precisely, they extend the target side CCG annotation in [Zollmann and Venugopal, 2006] to a tree-to-tree annotation. This bilingual grammar is then coarsened by grouping pairs of source and target labels into simpler categories according to a distance metric.

[9] Because string-to-tree systems do not require to match any input parse tree and the rules are assembled on the leaf non-terminals, decoding with string-to-tree SCFG or STSG works in the same way.

transformed into a target tree via the translation rules.

String-to-tree SCFG as described in [Williams and Koehn, 2012] is implemented in the Moses open source toolkit. The toolkit also implements a string-to-tree SCFG where hierarchical rules are augmented with syntactic labels on the target language side. Our evaluation (in Section 5.5) shows that the approach in [Williams and Koehn, 2012] outperforms all other syntax-based systems on several language pairs such as English-German.

Systems with source side syntactic annotations

Tree-to-string SCFG models [Hoang and Koehn, 2010], [Mylonakis and Sima'an, 2011] integrate syntactic annotations on the source side of SCFG rules. As noted in [Hoang and Koehn, 2010] these annotations restrict rule application in a way that hurts coverage. Consequently, tree-to-string models relax the source syntactic constraints. [Hoang and Koehn, 2010] propose a mixed-syntax model that combines hierarchical and tree-to-string rules. The authors report significant improvements over a hierarchical baseline. [Mylonakis and Sima'an, 2011] build a system using binary SCFG (see Section 2.2.1) with annotations on the source side of the rules. Similar to [Zollmann and Venugopal, 2006] they use CCG-style annotations. Early work on tree-to-string translation [Yamada and Knight, 2001] transforms a source language tree into a target string by performing a sequence of operations that model node reordering, insertion and translation.

The mixed syntax model of [Hoang and Koehn, 2010] is implemented in the Moses toolkit. The toolkit can also be used to build a system working with tree-to-string SCFG rules.

Binarization

Theoretically, the parsing complexity of SCFG decoding is exponential in the rank of the grammar, i.e. the maximal number of non-terminals in the source rhs of the rules. One way to reduce this complextiy is to reduce the rank of the grammar rules. While this is always possible with the (monolingual) CFG grammars, SCFG cannot always be binarized. [Zhang et al., 2006], [Huang et al., 2009] and [Xiao et al., 2009] present techniques to binarize some SCFG rules.

2.2.2 Weighted Synchronous Tree Substitution Grammars

Synchronous Tree Substitution Grammars (STSG) improve over SCFG by allowing multi-level node reordering (see Section 2.1.2). Many authors take advantage of this additional ability and build STSG based systems. Although STSG are useful when using grammars with syntactic annotations on both sides (tree-to-tree), this formalism has also been used to build systems with annotations on the source or target side only. None of these systems is publicly available so we could not experiment with these grammars in our work.

Tree-to-tree models

[Zhang et al., 2007] build an experimental SMT system using tree-to-tree STSG rules extracted from a small training set of 9,000 sentence pairs. They report significant improvements over a phrase-based system and very large improvements over a system based on tree-to-tree SCFG (as described in Section 2.2.1). Their evaluation also shows that tree-to-tree SCFG underperform phrase-based systems. Further work [Mi et al., 2008, Mi and Huang, 2008] and [Liu et al., 2009] point out that even when using STSG, tree-to-tree models are too constrained by the syntactic annotations. To overcome this problem [Mi et al., 2008] propose a system that takes a packed forest as input to the translation process instead of a sin-

gle tree. [Mi and Huang, 2008] extract translation rules from parse forests while [Liu et al., 2009] use these in the source side of tree-to-tree rules. All approaches improve over a standard (i.e using 1-best trees) tree-to-tree STSG system.

String-to-tree models

[Galley et al., 2004, Galley et al., 2006] present algorithms to extract and score string-to-tree STSG rules. [DeNeefe et al., 2007] extend this work by composing translation rules into larger units to improve lexical coverage. The authors show that string-to-tree STSG significantly improve over phrase-based systems [Koehn et al., 2003]. In turn, using composed rules yields better translation quality than minimal (i.e. not composed) ones. A comparison against other syntax-based approaches is not provided. [Marcu et al., 2006] build an SMT system on a restricted form of STSG rules where no non-terminals are allowed on the source side of the rules. They report significant improvements over phrase-based systems [Koehn et al., 2003]. An often outlined weakness of STSG rule extraction [Fossum et al., 2008, Cohn and Blunsom, 2009, Wang et al., 2010] is the bad compatibility between automatically acquired word alignments and parse trees: the alignments often cross constituents, which leads to the extraction of very large rules that do not generalize well. Many authors address this problem. [Wang et al., 2007] present a method to binarize string-to-tree grammar rules for better generalization and rule composition. Systems with binarized rules outperform non-binarized systems. [Wang et al., 2010] improve string-to-tree models by finding the tree structures, tree labels and alignments that best improve translation quality. Their approach significantly improves over standard string-to-tree systems. [Cohn and Blunsom, 2009] define a generative model for inducing string-to-tree STSG rules directly from target parsed parallel data. Translation with the obtained rules outperforms a standard string-to-tree system.

Tree-to-string models

[Huang et al., 2006] build a system based on a tree-to-string transducer (see Section 2.2.3) that can be seen as STSG-based translation with top-down decoding.[10] In contrast to most models presented above, language model scoring is not integrated in the decoding process and used as a reranker only. The approach is evaluated on a tiny test set of 140 sentences and underperforms a phrase-based system [Koehn et al., 2003]. [Liu et al., 2006] build a tree-to-string STSG system with bottom-up beam search decoding and integrated language model. The authors report significant improvements over a phrase-based system. [Liu et al., 2007] make tree-to-string rule application more flexible by allowing parse forests in the source side of the rules. This method significantly improves over a standard tree-to-string system.

2.2.3 Weighted Synchronous Tree Sequence Substitution Grammars

As seen in Section 2.1.3 synchronous tree sequence substitution grammars are even more flexible than STSG because they model translation rules containing sequences of trees instead of single ones. For these grammars, only tree-to-tree systems have been proposed so far.

Tree-to-tree models

[Zhang et al., 2008] build an SMT system with tree-to-tree weighted Synchronous Tree Sequence Substitution Grammars (STSSG). More precisely, they use the contiguous version of STSSG (Section 2.1.3) which requires that the tree sequences

[10] At each node, all possible translation rules are applied and descendant nodes are visited recursively. Decoding is made tractable by caching each visited node which guarantees that each node is visited at most once.

have contiguous spans.[11] Decoding is performed using a span-based bottom-up beam search. The authors show significant improvements over phrase-based systems as well as tree-to-tree SCFG and STSG based systems.[12]

[Sun et al., 2009] extend this work by using STSSG rules with tree sequences on (possibly) discontiguous spans (Section 2.1.3). Decoding is performed bottom-up in a three step heuristic for increasing spans. First, contiguous STSSG rules are applied to a specific span. In a second step, rules with discontiguous source sides are added to the candidates for this span. Finally, all (i.e. with and without source side discontiguities) rules with discontiguous target spans are applied. Language model scoring is only performed on contiguous rules. The authors evaluate the impact of different types of rules on translation quality. The results show that systems including non-contiguous rules outperform contiguous STSSG in all setups. Systems including source side discontiguities outperform systems with target side discontiguities alone.

Other Models

Further work on syntax-based SMT builds on formalisms other than the synchronous grammars discussed previously. [Ding and Palmer, 2005, Quirk et al., 2005, Riezler and Maxwell III, 2006] work with dependency grammars. [Graehl and Knight, 2004, Graehl et al., 2008, May et al., 2010] apply tree transducers to SMT problems while [Nesson et al., 2006, DeNeefe and Knight, 2009, Liu et al., 2011b] use Synchronous Tree Adjoining Grammars.

[11] In this approach, sequences are essentially used to overcome discrepancies between word alignments and parse trees.

[12] Their evaluation also confirms that systems based on tree-to-tree SCFG grammars perform poorly compared to all other evaluated systems.

2.3 Research Contributions

We contribute to the topic of SMT with synchronous grammars by building a system based on the Weighted Local Multi Bottom-Up Tree Transducer (l-MBOT, Section 2.1.3). More precisely, we work with the *Shallow l-MBOT* (Sh-l-MBOT), which is a restricted version of l-MBOT that works with shallow rules instead of tree fragments. Our contribution is presented in Chapter 5. It includes:

The definition of a Translation Model for Sh-l-MBOT. In Section 5.2, we present the Shallow l-MBOT translation model which is a log-linear model over Sh-l-MBOT derivations. The features include a gap penalty that counts the number of elements in the target side sequences of the rules.

A CYK-style bottom-up procedure for decoding with Shallow l-MBOT. The main difference between Shallow l-MBOT and synchronous SCFG rules are that the first allow sequences of trees on the target language side. We extended a bottom-up chart decoder [Chiang, 2007, Hoang, 2011] for synchronous SCFG grammars[13] to perform Sh-l-MBOT decoding. The language model is integrated in the decoder and cube pruning is performed for tractability. Our Sh-l-MBOT decoder integrates linguistic annotations at different levels:

The hierarchical Sh-l-MBOT decoder in Section 5.3 works without linguistic annotations and only exploits the structure of the Sh-l-MBOT grammar.

The tree-to-tree Sh-l-MBOT decoder in Section 5.4 integrates syntactic annotations on the source and target language side.

The string-to-tree Sh-l-MBOT decoder in Section 5.4.3 exploits target side annotations only.

A preliminary evaluation of the Sh-l-MBOT system. In Section 5.5, we evaluate the performance of a Sh-l-MBOT system with syntactic annotations on

[13]The formal description of Synchronous SCFG is given in Section 2.1.1.

the source and target language side (tree-to-tree). In this preliminary analysis, we show that tree-to-tree Sh-l-MBOT systems outperform SCFG based systems when the Sh-l-MBOT rules are combined with SCFG rules.

Our contributions base on the rule extraction procedures for Sh-l-MBOT grammars that have been proposed in [Maletti, 2011, Seemann et al., 2015a] which we briefly present in Section 5.1. A systematic evaluation of Sh-l-MBOT with syntactic annotations at different levels is performed in [Seemann et al., 2015b]. These results are presented in Section 5.5.

Closely related work is [Sun et al., 2009] who also experiment with STSSG that have discontiguities on one side of the translation rules (see Section 2.2.3). Our contributions are different from their work in many respects. First, Sh-l-MBOT translation rules allow only a single tree on the source side while they allow sequences of trees that have to be contiguous. Second, they only work with tree-to-tree rules while our system works with different setups of linguistic annotations. Moreover, we work with shallow rules while they use tree fragments. Third, their decoding algorithm only performs Language Model (LM) scoring on contiguous rules while we include LM scores for all rules. Finally, the evaluation of our systems in [Seemann et al., 2015b] leads to conclusions that are opposed to their observations: for tree-to-tree translation, target side discontiguities in Sh-l-MBOT systems hurt translation quality while [Sun et al., 2009] report significant improvements. A last difference is that our system scales to very large data sets (over 4 million parallel sentences) while they work on small data sets.

Chapter 3

Soft Syntactic Constraints

The hierarchical phrase-based SMT system [Chiang, 2005, Chiang, 2007] presented in Section 2.2.1 belongs to the best performing systems for several language pairs. Consequently, lots of research efforts have been dedicated to further improve hierarchical SMT. The main weakness of the hierarchical approach reported in the literature is the high ambiguity between the translation rules. Consequently, wrong rules are often applied during decoding which leads to bad translations. Approaches that try to solve this problem by replacing hierarchical rules with labeled SCFG rules, presented in Section 2.2, have been rather unsuccessful. Another way to improve hierarchical SMT is by augmenting the hierarchical translation model with syntactic information in the form of soft constraints or features. Many authors have followed this path and proposed different ways to include soft syntactic constraints in a hierarchical system. Most of these report significant improvements in translation quality for many language pairs. We contribute to this research by bringing forward further ways to integrate soft syntactic constraints into hierarchical decoding.

The present chapter is structured as follows: we begin by explaining the drawbacks of hierarchical rule application in Section 3.1 before discussing previous work on this topic in Section 3.2. We conclude by presenting our contributions in the field which we discuss in detail in Chapters 6 and 7.

3.1 Rule Application in Hierarchical SMT

The discussion of rule application in this section requires background knowledge in hierarchical machine translation. A detailed presentation of hierarchical systems is given in Chapter 4.

To present the main weakness in hierarchical rule application, let us consider rule r_7, which will be presented in Section 4.1.1. We call the string X_1 *de l' ingénérie* the source side of r_7 (instead of source right-hand side) and *engineering* X_1 its target side.

r_7 X $\rightarrow \langle$ X_1 de l' ingénérie, engineering X_1 $\rangle$

A consequence of the rule extraction heuristic in [Chiang, 2007], that will be introduced in Section 4.1.1, is that the obtained hierarchical grammar contains numerous rules with the same source language side (but different target language sides). For example, rules q_1 and q_2 have the same source language side X_1 *pratique* X_2 although their target language sides *practical* X_1 X_2 and X_1 X_2 *process* are different.

q_1 X $\rightarrow \langle$ X_1 pratique X_2, practical X_1 X_2 $\rangle$
q_2 X $\rightarrow \langle$ X_1 pratique X_2, X_1 X_2 process $\rangle$

During decoding (see Section 4.2), rules with the same source side apply to the same segments in the input sentence: both rules q_1 and q_2 can be used for translating sentences F_1 and F_2 below:[1]

F_1 Une étude de l' (intérêt)$_{X_1}$ **pratique** (de notre approche)$_{X_2}$.
A study on the (interest)$_{X_1}$ ***practical*** *(of our approach)*$_{X_2}$.
F_2 Une étude de (la)$_{X_1}$ **pratique** (de l'ingénérie informatique)$_{X_2}$.
A study on (the)$_{X_1}$ ***process*** *(of software development)*$_{X_2}$.

However, among all applicable rules, only a subset yield correct translations. For instance, using q_2 to translate F_1 yields the incorrect translation E_2 while the correct

[1] Given with the corresponding English word-by-word glosses.

translation E_1 requires the application of q_1.

E_1 A study on the **practical** (interest)$_{X_1}$ (of our approach)$_{X_2}$.
E_2 *A study on the (interest)$_{X_1}$ (of our approach)$_{X_2}$ **process**.

On the other hand, the application of q_1 to translate F_2 yields the incorrect translation E_3 while the correct translation E_4 requires to apply rule r_2.

E_3 *A study on **practical** (the)$_{X_1}$ (software development)$_{X_2}$.
E_4 A study on (the)$_{X_1}$ (software development)$_{X_2}$ **process**.

The relative frequency based scoring heuristic for hierarchical rules [Chiang, 2007] (see Section 4.1.2) does not allow a good disambiguation between rules with the same source side. For instance, in scientific reports or newswires, parallel sentences from which rules q_1 and q_2 can be extracted are equally likely to be observed. These rules will then have about the same probability after rule scoring. As a consequence, the rule weight feature of a standard hierarchical system will allow a random selection of q_1 or q_2 to translate sentences such as F_1 and F_2. Furthermore, if rule q_1 has been extracted more frequently from the training data than q_2 but a sentence similar to F_2 appears in the test data, then the rule weight feature will erroneously boost the application of rule q_1.

3.2 Better Rule Application with Syntactic Features

Several approaches have been proposed to improve the application of hierarchical rules using syntactic information. A first line of work [Vilar et al., 2008] extends the hierarchical rule extraction heuristic to determine if the non-terminals in the right-hand side (rhs) of hierarchical rules match a syntactic constituent. Using this information, a syntactic score is computed which is added to the rule features presented in Section 4.1.2. Besides this work, many approaches focus on the context of the source sentence in which a hierarchical rule is applied. Given $r : X \rightarrow \langle \gamma, \alpha, \sim \rangle$,

previous work on rule selection can be grouped into 2 categories, which we present in detail in the next sections:

1. Models that estimate how good rule application is in a given situation by considering the syntactic structure of the source and/or target language sentence to which the rule is applied. Approaches considering source language syntax only aim to estimate a distribution $P(r \mid S_s)$, where S_s denotes the source syntax. Work taking the target language sentence into account aim to estimate $P(r \mid S_s, S_t)$, where S_t denotes the target syntax. We call such models *Syntactic context models*.

2. Models that estimate how good the application of the target side γ of a rule is given the source side α as well as information about (i) the structure of the rule to be applied and (ii) the syntactic structure of the source language sentence to which the rule is applied. Such models aim to estimate a distribution that can be written $P(\gamma \mid \alpha, S)$.[2] Following, [He et al., 2008] we call such models *Syntactic rule selection models*.

Finally, several authors use syntactic structure to train reordering models. [Gao et al., 2011] uses soft syntactic constraints from the dependency structure of the source sentence to predict the reordering between a word and its head. [Li et al., 2014] propose a unified reordering model that uses syntactic and semantic information to predict constituent reordering.

3.2.1 Syntactic Context Models

The first attempt to improve hierarchical rule selection by considering information about the syntactic structure of the source sentence is [Chiang, 2005, Chiang, 2007]. This approach rewards hierarchical rules whose span matches syntactic constituents in the parse tree of the source language sentence. The rewarding

[2] Where S denotes the information about the structure of the rules and the syntax of the source sentence.

is done with a binary feature $c(i,j)$ added to the hierarchical translation model. This approach did not lead to improvements in translation quality. Further work [Marton and Resnik, 2008, Marton et al., 2012] extend the constituency feature $c(i,j)$ into a finer grained set of features $C = \{NP, VP, IP, ...\}$ that includes the types of constituents that are matched. In addition to rewarding constituent matching, crossing is penalized. In [Marton and Resnik, 2008] these features are called *soft syntactic constraints*. The inclusion of these in the hierarchical translation model leads to significant improvements in translation quality. [Chiang, 2010] extends this work by augmenting a hierarchical system not only with syntactic features on the source sentence but also on the target sentence, which is built during decoding. [Liu et al., 2011a] define a discriminative model over source side constituent labels instead of using rewarding/crossing heuristics. The training procedure for their discriminative model is based on possible derivations on the source sentence.[3] Both approaches [Chiang, 2010, Liu et al., 2011a] yield significant improvements in translation quality.

3.2.2 Syntactic Rule Selection Models

Hierarchical rule selection models are inspired by work on word sense disambiguation (WSD) [Carpuat and Wu, 2007]. The first attempt to integrate a WSD model into a hierarchical system is [Chan et al., 2007]. Their model is limited to rules containing only terminal symbols and having length 2. This work is extended by [He et al., 2008] who consider all hierarchical rules and build a discriminative model which they call *rule selection model*. This model includes features on the structure of hierarchical rules as well as contextual information about the source sentence. Following [Carpuat and Wu, 2007, Chan et al., 2007], they define a rule selection model that is local to the source side of the rules, i.e. that trains one (local) classifier for each source side or pattern of the hierarchical rules. In

[3]The training instances are obtained by performing bilingual parsing on the training data and extracting the obtained rules from the derivation forest.

[He et al., 2010] they somewhat generalize this work by defining a model that is local to source patterns instead of rule source sides. This first line of work on rule selection does not include syntactic features and focuses on lexical rule selection. The first true *Syntactic rule selection model* is proposed in [Cui et al., 2010] who define a model that includes features on the syntactic structure of the source sentence. Instead of limiting their work to a rule selection model, they propose a joint approach over the source and target side of hierarchical rules. This global model includes a *Syntactic rule selection model* which they call the *Context Based Target Model*. The model in [Cui et al., 2010] is not local to the source side of the rules but generalizes to all rules (global model). Syntactic rule selection models are also defined to improve other types of syntax- and even semantics-based systems. [Liu et al., 2008] train a local discriminative rule selection model for tree-to-string machine translation. [Zhai et al., 2013] propose a discriminative model to disambiguate predicate argument structures. [Huang et al., 2010] propose a model that includes information about the syntactic structure of (i) the source sentence and (ii) the non-terminals in the lhs and rhs of the hierarchical rules. Each rule is augmented with a real-valued vector of syntactic features. Each entry represents a syntactic constituent that has been observed to govern the source rhs non-terminal X in the training data. This work leads to significant improvements on an English-to-German translation task.

3.3 Syntactic Rule Selection for String-to-Tree SMT

As seen in Section 3.2.1, syntactic rule selection models have shown significant improvements in hierarchical, tree-to-string and predicate augment structure based systems. A common point between these systems is that they have no syntactic annotations on the target language side. An interesting research question is then if such models can also be applied to systems with target syntactic annotations such as the string-to-tree system presented in Section 2.2.1. Because they beat all

other syntax-based systems on several language pairs [Bojar et al., 2014], string-to-tree systems are particularly good candidates for testing a syntactic rule selection model.[4] Rules q_3 to q_6 below are string-to-tree rules as presented in Section 2.2.1. These rules have in common that they have the same source rhs and are thus potential candidates to translate sentence F below.

(q_3) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, $JJ_1\ JJ_2$ characteristics $\rangle$
(q_4) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, NNS_1 characteristic $JJ_2\ \rangle$
(q_5) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, JJ_1 properties $JJ_2\ \rangle$
(q_6) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, $JJ_1\ JJ_2$ features $\rangle$

A syntactic rule selection model for string-to-tree systems should predict, during decoding[5], which rule should be applied to produce the correct translation E, i.e. rule q_3.

F (Diverses)$_{X1}$ caractéristiques (importantes)$_{X2}$ n'ont pas été prises en compte.
(Various)$_{X_1}$ ***characteristics*** *(important)$_{X_2}$ were not considered.*
E (Various)$_{X1}$ (important)$_{X_2}$ **characteristics** were not considered.

No work has yet been proposed on integrating a syntactic rule selection model into a string-to-tree system. One of our contributions (see Section 3.4) is to define, implement and evaluate such a system.

3.4 Research Contributions

Our first contribution to the topic of improving SMT with soft syntactic constraints is the definition of a global syntactic rule selection model for hierarchical SMT. This contribution is presented in Chapter 6 and includes:

[4] As shown in Section 2.2, string-to-tree systems can either be based on synchronous context-free grammars or synchronous tree substitution grammars. We work with the first because it is implemented in Moses which is publicly available. Another advantage of using Moses is that this system has been evaluated on shared tasks and shown to achieve very competitive performance.

[5] SCFG decoding with target side annotation is explained in Section 4.4.4.

The definition and training of a global rule selection model. In Section 6.2, we define and train a syntactic rule selection model that generalizes to the complete hierarchical grammar instead of rules with the same source language side only. The model includes a rich set of syntactic and lexical features. By training the model with a high-speed streaming classifier we can train a global model without performing any pruning.

The integration of the model in a hierarchical system. In Section 6.4, we present to which extent the hierarchical decoding procedure[6] had to be modified to include model predictions during decoding.

An extensive evaluation. In Section 6.5, we show the advantages of using a global model without pruning over previously proposed approaches. These observations are confirmed by an extensive empirical evaluation, which we present in Section 6.6.

Closely related work to our first contribution is [Cui et al., 2010] who also define a global syntactic rule selection model for hierarchical systems. Our contribution is different from their work insofar as it allows the training of a global model without pruning of negative examples (see Section 6.5.2).

Our second contribution to this field is the extension of the hierarchical syntactic rule selection model to a string-to-tree system. This contribution is presented in Chapter 7 and includes:

A syntactic rule selection model for string-to-tree SMT. In Section 7.2, we define and train a syntactic rule selection model for string-to-tree SMT. The model is global and hence generalizes to the complete string-to-tree grammar. The features of this model include rule shape features and soft syntactic constraints of the source sentence.

[6] Presented in Section 4.2.

A preliminary evaluation of the model. In Section 7.3, we evaluate a string-to-tree system augmented with the syntactic rule selection model. These results are further analyzed and compared to a hierarchical system.

Chapter 4

Background: Decoding for Synchronous Context-Free Grammars

Our contributions to research in Statistical Machine Translation (SMT), presented in Sections 2.3 and 3.4, create improved SMT systems either by implementing novel grammar formalisms or by integrating soft syntactic constaints in existing translation models. A central aspect in building these systems is the implementation of customized decoding procedures. The decoders presented in our work are based on the bottom-up chart decoder for Synchonous Context-Free Grammars (SCFG) implemented in Moses [Hoang et al., 2009] and described in [Chiang, 2007, Hoang, 2011]. The goal of this chapter is to provide the reader with background knowledge in decoding for SCFG that is necessary to fully understand our contributions.

We begin with a brief presentation of hierarchical grammars and the hierarchical translation model in Section 4.1. Then we provide a detailed description of the hierarchical decoding procedure (as implemented in Moses [Hoang et al., 2009]) in Section 4.2. Finally, we extend our presentation to systems dealing with syntactically annotated SCFG in Section 4.3.

4.1 Hierarchical SMT

4.1.1 Hierarchical Grammar

Hierarchical Rules

A Hierarchical Grammar (HG) is a Synchronous Context-Free Grammar (SCFG), as defined in Chapter 2 (Section 4.3) with two specificities. First, their input and output alphabets N_s and N_t only contain the labels X and S. Moreover HG include a set of glue rules that sequentially combine non-terminals to the start symbol. Following [Chiang, 2007], we denote hierarchical rules by $X \rightarrow \langle \alpha, \gamma, \sim \rangle$, where α and γ are the source and target language strings and $\sim$ the correspondence between non-terminal symbols. Usually, we show the alignment $\sim$ by indexing the non-terminals in the rules. Example hierarchical rules are h_1 to h_5 below.

h_1 $X \rightarrow \langle\ X_1\ X_2,\ X_2\ X_1\ \rangle$

h_2 $X \rightarrow \langle$ une $X_1\ X_2,$ a $X_2\ X_1\ \rangle$

h_3 $X \rightarrow \langle$ une X_1 pratique, a practical $X_1\ \rangle$

h_4 $X \rightarrow \langle$ pratique, practical $\rangle$

h_5 $X \rightarrow \langle$ étude, study $\rangle$

In addition to the SCFG rules above, HG include two glue rules that have the form:

g_1 $S \rightarrow \langle\ S_1\ X_2,\ S_1\ X_2\ \rangle$

g_2 $S \rightarrow \langle\ X,\ X\ \rangle$

The semantics of hierarchical grammars is the same as for the corresponding SCFG including the glue rules. Example D below shows a hierarchical derivation:

D $S \underset{g_2}{\Rightarrow} \langle\ X_1, X_1\ \rangle \underset{h_2}{\Rightarrow} \langle$ *une* $X_1\ X_2$, *a* $X_2\ X_1\ \rangle \underset{h_4}{\Rightarrow} \langle$ *une* X_1 *pratique, a practical* $X_1\ \rangle$
$\underset{h_5}{\Rightarrow} \langle$ *une étude pratique, a practical study* $\rangle$

Automatic Extraction of Hierarchical Rules

The extraction of hierarchical rules [Chiang, 2007] is done following a two-step heuristic. The input to rule extraction is a word aligned parallel text. The sentences F and E show example sentences from a French-English parallel corpus. Figure 4.1 shows a possible word alignment between sentences F and E.

F Une étude de la pratique de l'ingénérie informatique.

E A study of the software engineering process.

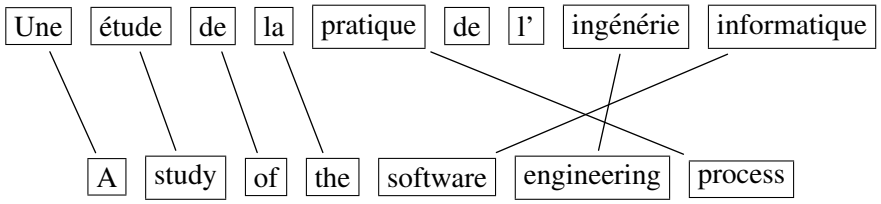


Figure 4.1: Word aligned sentences

In the first step, the rule extraction yields "initial phrase pairs" [Chiang, 2007, p.266]. Given a word aligned parallel text like the one in the above figure, a pair of token sequences $\langle f_i^j, e_{i'}^{j'} \rangle$ is an initial phrase pair if it fulfills the conditions below. The sequences f_i^j and $e_{i'}^{j'}$ are strings of terminal symbols in the source and target language that span from i to j. We use the symbol $\sim$ to denote word alignment.

1. There is at least one aligned word between both phrases composing the initial phrase pair.

   Formally, $f_k \sim e_{k'}$ for some $k \in [i, j]$ and $k' \in [i', j']$.

2. All words aligned to the source language phrase are in the target language phrase.

   Formally, $f_k \nsim e_{k'}$ for all $k \in [i, j]$ and $k' \notin [i', j']$.

3. All words aligned to the target language phrase are in the source language phrase.

Formally, $f_k \nsim e_{k'}$ for all $k \notin [i, j]$ and $k' \in [i', j']$.

All initial phrase pairs are hierarchical rules without non-terminals on the right-hand side, i.e. rules of the form $X \rightarrow \langle f_i^j, e_{i'}^{j'} \rangle$. Further hierarchical rules are created from initial phrase pairs by removing subpairs and replacing those by pairs of aligned non-terminals. Formally, for each rule $r = X \rightarrow \langle \alpha, \gamma \rangle$, an initial phrase pair $\langle f_i^j, e_{i'}^{j'} \rangle$ can be extracted from r if $\alpha = \alpha_1 f_i^j \alpha_2$ and $\gamma = \gamma_1 e_{i'}^{j'} \gamma_2$. The extracted rule is given by $X \rightarrow \langle \alpha_1 X_k \alpha_2, \gamma_1 X_k \gamma_2 \rangle$ for a new index k.

To illustrate the extraction of hierarchical rules, consider the word aligned sequences in Figure 4.2. Initial phrase pairs extracted from this sequence include[1]:

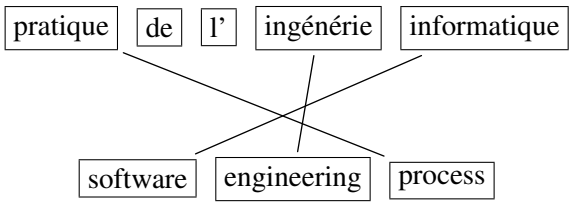


Figure 4.2: Word aligned noun phrases

r_1 $X \rightarrow \langle$ pratique, process $\rangle$

r_2 $X \rightarrow \langle$ pratique de, process $\rangle$

r_3 $X \rightarrow \langle$ pratique de l', process $\rangle$

r_4 $X \rightarrow \langle$ ingénérie, engineering $\rangle$

r_5 $X \rightarrow \langle$ pratique de l' ingénérie, engineering process$\rangle$

r_6 $X \rightarrow \langle$ pratique de l' ingénérie informatique, software engineering process$\rangle$

Further rules are created by subtracting initial phrase pairs from larger phrase pairs. For instance, a new rule r_7 can be created by subtracting the right-hand side (rhs) of rule r_1, that is the initial pair $\langle$ pratique, process $\rangle$ from rule r_5. In the same fashion, rule r_8 is obtained by subtracting the rhs of rule r_2 from r_5.

r_7 $X \rightarrow \langle$ X_1 de l' ingénérie, engineering X_1 $\rangle$

r_8 $X \rightarrow \langle$ X_1 l' ingénérie, engineering X_1 $\rangle$

[1]As the rule extraction heuristic leads to a large number of extracted initial phrase pairs, we only give some example rules here.

By subtracting several initial phrase pairs from hierarchical rules, new rules containing multiple non-terminals are created. For instance, rule r_9, below, is created by subtracting the rhs of r_1 and r_4 from r_5. In the same fashion, rule r_{10} is created by subtracting rules r_3 and r_4 from r_5.

r_9 $X \rightarrow \langle\ X_1$ de l' X_2, $X_2\ X_1\ \rangle$

r_{10} $X \rightarrow \langle\ X_1\ X_2$, $X_2\ X_1\ \rangle$

Because this rule extraction strategy leads to a very high number of rules, [Chiang, 2005] adds the following restrictions to the extraction heuristic:

1. Only the shortest of several initial phrase pairs with the same alignment points is kept.
2. Initial phrase pairs are limited to a length of 10 symbols on the source language side. Rules are limited to five[2] symbols on the source language side.
3. The subtracted initial phrase on the source language side must contain more than one word.
4. Rules can have at most 2 non-terminals.
5. No rule with adjacent non-terminals on the source language side is allowed.
6. Rules must have at least one pair of aligned terminals.

By applying this restriction to the extraction heuristic illustrated above, rules r_2 and r_3 would not be extracted because they are not the smallest rules with the same alignment points (restriction 1). Consequently, rule r_8 would not be extracted because it requires to subtract the initial phrase pair r_2, which is not allowed (restriction 1). Rule r_7 would not be extracted because the subtracted phrase pair, i.e. the rhs of rule r_1, contains only one symbol on the source language side (restriction 4). Rule r_9 would also be dropped because it does not contain a pair of aligned terminals (restriction 7). Finally, rule r_{10} would not be extracted because its source language side contains adjacent non-terminals (restriction 6).

[2]terminal and non-terminal

4.1.2 The Hierarchical Translation Model

Many hierarchical rules can be applied to an input sentence (e.g. Sentence F in Section 4.1.1) to derive output sentences and each derivation possibly[3] yields a different output sentence. The hierarchical translation model predicts the best translation e of a given input sentence f.

Mathematical Definition

The hierarchical translation model is a log-linear model [Och and Ney, 2002] over the derivations D. The probability $P(D)$ of a derivation $D = S \underset{r_1}{\Rightarrow} \langle \alpha_1, \gamma_1 \rangle \underset{r_2}{\Rightarrow} ... \underset{r_l}{\Rightarrow} \langle f, e \rangle$ is defined as the weighted product of the features in this derivation.

$$P(D) \propto \prod_{i=1}^{m} h_i(D)^{\lambda_i} \tag{4.1}$$

The features $h_i(D)^{\lambda_i}$ belong to two categories which are (i) features on the rules and (ii) the n-gram language model. Considering this division, Equation 4.1 can be written as:

$$P(D) \propto LM(e)^{\lambda_m} \prod_{i=1}^{m-1} h_i(D)^{\lambda_i} \tag{4.2}$$

where $LM(e)$ is the evaluation of the language model on e. We assume that the language model is the m-th feature. As seen in Section 2.1.1, the product of the rules in a SCFG derivation equals the weight of this derivation. Hence, Equation 4.2 can be written as:

$$P(D) \propto LM(e)^{\lambda_m} \prod_{i=1}^{m-1} \prod_{j=1}^{l} h_i(r_j)^{\lambda_i} \tag{4.3}$$

[3]Note that different derivations can yield the same output sentence. Given the input sentence *pratique de l'ingénérie*, the derivation using rules r_1 and r_7 produces the same output sentence as the derivation using rules r_2 and r_8.

Model Features

As seen above (Equation 4.3), the features of the hierarchical translation model are the target language model $LM(e)$ as well as a set of features on the rules in a derivation. Rule features include:

1. The direct translation weight of a rule, which defines the probability $P(\gamma \mid \alpha, \sim)$ to see the target side γ of a rule given its source side α.
2. The indirect translation weight of a rule, which defines the probability $P(\alpha \mid \gamma, \sim)$ to see the source side α of a rule given its target side γ.
3. The direct lexical weight of a rule, which defines the probability $P_w(u \mid v)$ to see the words u in the target side of a rule given the words v in the source side.
4. The indirect lexical weight of a rule, which defines the probability $P_w(v \mid u)$ to see the words v in the source side of a rule given the words u in the target side.
5. A constant used to count the number of rules in a derivation.
6. A constant used to count the number of glue rules.
7. The number of terminal symbols in a rule.

Feature Training

Translation weights To score hierarchical rules according to a generative model, the derivations from which the sentence pairs composing the training data have been obtained should be known. Using this information, observed rules could be counted to estimate a distribution. However, this information is not available and typical training data consists of word aligned sentence pairs as illustrated in Section 4.1.1. As a consequence, [Chiang, 2007] proposes to score hierarchical rules in a similar fashion as in phrase-based systems [Koehn et al., 2003], by computing relative frequencies on extracted rules. Each time an initial phrase pair is extracted, it gets a count of 1. This count is then uniformly divided to obtain fractional counts allocated to hierarchical rules that can be obtained from this initial phrase-pair. The frequency of each hierarchical rule is obtained by summing up the fractional counts of rules extracted from the training data. The direct probability is then obtained by

normalizing its frequency by the (fractional) count of all rules with the same source language side, i.e. by computing $P(\gamma \mid \alpha, \sim) = \frac{fracCount(\alpha,\gamma)}{\sum_{\gamma_i} fracCount(\alpha,\gamma_i)}$. In the same fashion, the indirect translation probability is obtained by normalizing over rules with the same target language side i.e. by computing $P(\alpha \mid \gamma, \sim) = \frac{fracCount(\alpha,\gamma)}{\sum_{\alpha_i} fracCount(\alpha_i,\gamma)}$. In order to illustrate the scoring of hierarchical rules, assume that the initial phrase pair r_6 has been extracted 20 times from the aligned corpus. According to the restricted hierarchical rule extraction heuristic, rules r_{11} and r_{12} below can be created from r_6.

r_{11} $X \rightarrow \langle\ X_1$ informatique, software $X_1\ \rangle$ 10

r_{12} $X \rightarrow \langle$ pratique de l'X_1, X_1 process $\rangle$ 10

Each count attributed to r_6 is uniformly distributed among r_{11} and r_{12}, meaning that each rule gets a frequency of 10. The frequency of each rule is indicated in red. Assume that r_{13} to r_{16} have also been extracted from the training data:

r_{13} $X \rightarrow \langle$ pratique de l'X_1, practical of $X_1\ \rangle$ $\frac{15}{2}$

r_{14} $X \rightarrow \langle$ pratique de l'X_1, X_1 practice $\rangle$ 20

r_{15} $X \rightarrow \langle$ pratique de l'X_1, X_1 in practice $\rangle$ $\frac{25}{2}$

r_{16} $X \rightarrow \langle$ processus X_1, X_1 process $\rangle$ 10

The direct probability $P(\gamma_{12} \mid \alpha_{12})$ of rule r_{12} is then computed by dividing its frequency by the sum of the frequencies of the rules having the segment *pratique de l'*X_1 as source language side, i.e. by the sum of the frequencies of r_{12} to r_{15}. According to the example above, we have $P(\gamma_{12} \mid \alpha_{12}) = \frac{10}{\frac{15}{2}+20+\frac{25}{2}+10} = \frac{10}{50} = \frac{1}{5}$. The indirect translation probability is obtained through division by the rules having the segment X_1 *process* as target language side. This yields $P(\alpha_{12} \mid \gamma_{12}) = \frac{10}{20} = \frac{1}{2}$.

Lexical weights Lexical weights are obtained by multiplying the lexical probabilities of aligned words in a rule. If a word is aligned to multiple words, the average of the lexical probabilities is taken. The computation of the direct lexical weight is given by $P_w(u \mid v) = \prod_{i=1}^{|u|} \frac{1}{|\{j|(i,j)\in A\}|} \sum_{(i,j)\in A} lex(u_i \mid v_j)$, where (i) v and u denote the string of the terminal symbols in α and γ, (ii) A stands for

$v \sim u$, which is the (NULL-enriched) alignment between the words in α and γ, (iii) $lex(u_i \mid v_j)$ denotes the word translation probability between the j-th word in v and the i-th word in u. In the same fashion, the indirect lexical weight is computed as $P_w(v \mid u) = \prod_{j=1}^{|v|} \frac{1}{|\{i|(i,j)\in A\}|} \sum_{(i,j)\in A} lex(v_j \mid u_i)$. Unaligned words are considered as being aligned to a special empty word which also has a given lexical translation probability. To illustrate the training of lexical weights, consider rules r_6 and r_{17}. The alignments between terminal symbols in these rules are given in Figure 4.3 and the word translation probabilities between aligned words are given in Figure 4.5. The direct lexical weight of r_6 is $P_w(u_6 \mid v_6) = \frac{0.15+0.05}{2} * 0.4 * 0.5 * 0.3 = 0.006$. The indirect lexical weight of r_6 is $P_w(v_6 \mid u_6) = 0.3 * 0.01 * 0.01 * 0.5 * 0.2 = 3 * 10^{-6}$. Computed in the same way, the direct lexical weight of r_{17} is $P_w(u_{17} \mid v_{17}) = \frac{0.15+0.1+0.05}{3} = 0.1$. The indirect weight is $P_w(v_{17} \mid u_{17}) = 0.05 * 0.1 * 0.05 = 0.00025$

r_{17} X $\rightarrow \langle$ compte tenu de X_1, considering X_1 $\rangle$

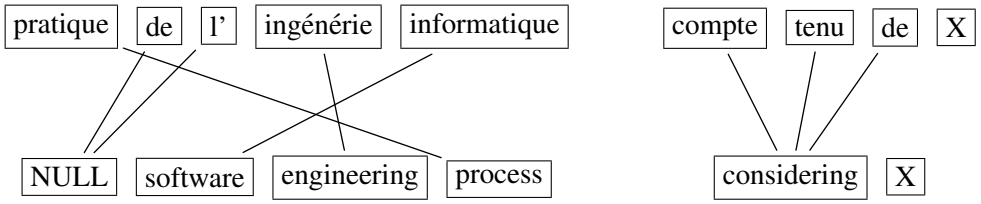


Figure 4.3: Alignment between words in rules r_6 (left) and r_{17} (right)

Count features The count features assign a negative weight or penalty of e^{-1} to the elements they are defined on. An example count feature is the number of rules

French Word	English Word	Weight
pratique	process	0.4
de	NULL	0.15
l'	NULL	0.05
ingénérie	engineering	0.5
informatique	software	0.3

French Word	English Word	Weight
compte	considering	0.15
tenu	considering	0.1
de	considering	0.05

Figure 4.4: Direct word translation probabilities of aligned words in rules r_6 (left) and r_{17} (right)

English Word	French Word	Weight
process	pratique	0.3
NULL	de	0.01
NULL	l’	0.01
engineering	ingénérie	0.5
software	informatique	0.2

English Word	French Word	Weight
considering	compte	0.05
considering	tenu	0.1
considering	de	0.05

Figure 4.5: Indirect word translation probabilities of aligned words in rules r_6 (left) and r_{17} (right)

used in a derivation.

Weight training The weights λ_i of the features in the hierarchical translation model are trained using minimum error rate training [Och, 2003].

4.2 Hierarchical Decoding

The decoding algorithm of a hierarchical SMT system aims to find the yield e of the 1-best derivation obtained by applying hierarchical rules on the input string f, that is:

$$\hat{D} = \arg\max_{D} P(D) \tag{4.4}$$

where $P(D)$ is given by Equation 4.3 and $D = S \underset{r_1}{\Rightarrow} \langle \alpha_1, \gamma_1 \rangle \underset{r_2}{\Rightarrow} ... \underset{r_l}{\Rightarrow} \langle f, e \rangle$. The generated target translation $(\hat{e})$ of the best derivation is the proposed translation.

The 1-best derivation $\hat{D}$ for a French string f is obtained by parsing f with the source side of the hierarchical grammar. This is done, as for CFG grammar rules[4] in Chomsky normal form, with the CYK chart parser. In order to avoid the conversion of the rules into Chomsky normal form, [Chiang, 2007] and [Hoang, 2011] use parsing algorithms that operate directly on the grammar rules. Since they use different methods, we follow [Hoang, 2011], whose work is implemented in the Moses open source toolkit [Koehn et al., 2007, Hoang et al., 2009].[5] This search

[4]An overview of context-free grammars (CFG) has been given in Section 2.1.1.

[5]We do this because our decoder for the shallow l-MBOT translation model is an extension of

procedure is based on the CYK+ chart parser [Chappelier et al., 1998]. Once the parsing process yielded the best derivation $\hat{D}$, the corresponding target translation $(\hat{e})$ must be produced. This is done by adding a translation generation component to the CYK+ chart parser[6]. Finally, the score $LM(e)^{\lambda_m}$ of the language model (in Equation 4.3) must be integrated in the parsing procedure. As this integration makes search expensive, pruning is applied to make the decoding process tractable.

We first present the CYK+ parsing algorithm in Section 4.2.1. Then we introduce the translation generation component (in Section 4.2.2) before discussing how n-best lists are generated for minimum error rate training (in Section 4.2.4). We close this section by showing how language model scores are computed (in Section 4.2.5) and by presenting pruning (in Section 4.2.6).

4.2.1 The CYK+ parsing algorithm

[Chiang, 2007, Hoang, 2011] formalize the CYK+ parsing algorithm as a deductive proof system and define a decoding procedure using this system. We begin by introducing deductive proof systems. Then we present the inference rules of the chart parser before showing the search procedure for an input sentence. As we present the CYK+ parser for hierarchical systems, we take all non-terminal symbols from the alphabet $N_s = N_t = \{X, S\}$, as defined in Section 4.1.1.

Deductive Proof Systems

A deductive proof system consists of (i) a set of weighted items, which we write $X_i : w_i$, and (ii) a set of inference rules of the form:

$$\frac{A_1 : w_1 \cdots A_k : w_k}{B : w} \quad \phi$$

Moses.

[6]Note that when searching for the first best derivation only (and without scoring the language model) the generation of the English string e can easily be done as a postprocessing step that replaces each source side α of a rule with its target side γ.

where $A_i : w_i$ and $B : w$ are weighted items and ϕ is a side condition. Inference rules are used in the inference process to prove new items given a set of items that have already been proved. The first, denoted by $B : w$ in the inference rule, is called the *consequent* of the rule. The latter, written $A_i : w_i$, are called the *antecedents*. The inference process starts with a set of items, called the *axioms*, which are assumed being already proved (or true) before inference starts. Given these axioms and the inference rules, new items are proved. The process stops once a special axiom, called the *goal* of the process, has been proved.

The CYK+ Chart Parser

Instead of using grammar rules in Chomsky normal form, the CYK+ chart parser uses dotted rules of the form:

$$X \rightarrow \alpha_1 \bullet \alpha_2$$

where α_1 and α_2 are strings composed of terminal or non-terminal symbols. Dotted rules are applied on the input string in a bottom-up fashion, just as for the CYK algorithm. In addition, the rules memorize the processed symbols in the right-hand side (rhs) of a rule by moving the dot left-to-right, one symbol at a time. Once the dot has consumed all symbols in the rhs of a rule, this rule is called *passive*. Passive rules can be used to parse non-terminal symbols in the same fashion as grammar rules in the CYK parsing algorithm.

Written as a deductive proof system, the CYK+ chart parsing algorithm is composed of items of the form $[X \rightarrow \alpha_1 \bullet \alpha_2, i, j, w]$ where $X \rightarrow \alpha_1 \bullet \alpha_2$ is a dotted rule in which the string α_1 has been recognized. The complete item indicates that the string α_1 spanning from i to j has been processed inside of a subtree rooted by X. For the complete subtree to be processed, the string α_2 remains to be recognized. The last element of the item is the weight w of the subtree. We call such items *parse items*. We distinguish between *active* and *passive* parse items. The first are composed of active dotted rules and the second of passive rules. The axioms

of the parser are obtained by means of the grammar rules $X \xrightarrow{w} \alpha$. For each rule, the axiom in Equation 4.5 is created. Because axioms model rules that have not yet been applied, we omit the span information in the remainder of the thesis.

$$\frac{}{[X \rightarrow \bullet\alpha, i, i, w]} \quad (X \xrightarrow{w} \alpha \in P) \tag{4.5}$$

The axiom indicates that the string α, that is the rhs of the grammar rule $X \xrightarrow{w} \alpha$, remains to be recognized for the subtree rooted by X to be processed. The inference rules of the parser are divided into two categories:

1. Lexical rules process terminal symbols and are of the form:

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w]} \tag{4.6}$$

   where t_{j+1} is the terminal spanning from j to $j+1$. The antecedent of the rule is a parse item where the terminal symbol t_{j+1} has not yet been recognized. The consequent of the rule is an item in which t_{j+1} has been processed and the span extended to $j+1$.

2. Non-lexical rules process non-terminal symbols and have the form:

$$\frac{[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1] \quad [X \rightarrow \beta\bullet, j, k, w_2]}{[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2]} \tag{4.7}$$

   where X is a non-terminal spanning from j to k. The first antecedent of the rule is a parse item where the non-terminal X remains to be processed. The second antecedent is a parse item where the rule $X \rightarrow \beta\bullet$ is passive, meaning that its right hand side has already been consumed. The consequent is a parse item where the non-terminal X has been processed via the passive rule $X \rightarrow \beta\bullet$ and the span extended from j to k.

Finally the goal of the system is the item $[S \rightarrow \alpha\bullet, 0, |f|, w]$, where the rhs of a rule starting with the start non-terminal S and spanning the entire input string has

been processed.

CYK+ search Procedure

The CYK+ parsing process starts with the axioms defined above and produces new items using the inference rules. The search procedure is given in Algorithm 1. It uses six data structures. The first two, denoted by $a[\cdots]$, are lists containing the axioms constructed using the rules of the grammar, one list for each non-terminal of the grammar[7]. The second and fourth, denoted by $d[\cdots]$, contain active parse items for each non-terminal. The third and fifth, indicated by $h[\cdots]$, contain passive parse items for each non-terminal. The item lists are ordered from smallest to largest span in an input sentence f of length $|f|$.

The parsing algorithm traverses each list cell by cell. In each cell, the parser tries to prove all items belonging to this cell. Each time a new item is proven, it is added to the cell together with a tuple of back pointers to the items from which it has been inferred. Proven items that are still active are added to the list of active items (line 7). Proven items that are passive are added to the list of passive items (line 10). Once all items rooted by X have been handled, the parser processes the items rooted by S. When several equivalent items populate a cell, the one with the best weight is kept, together with its back pointers. When the complete input sentence has been processed, the parsing procedure ends. If the goal $[S \rightarrow \alpha\bullet, 0, |f|, w]$ has been proved, the sentence f could be parsed successfully. The best derivation is obtained by starting with the goal item $[S \rightarrow \alpha\bullet, 0, |f|, w]$ and following the back pointers stored in each item.

[7] [Chiang, 2007] puts all rules in the same list. Because X rules have to be applied before S rules, we found it convenient to put each rule type in a specific list.

Algorithm 1 CYK+ search algorithm

Data structures:
- $a[X]$: list of axioms created from rules $X \xrightarrow{w} \alpha \in P$
- $a[S]$: list of axioms created from rules $S \xrightarrow{w} \alpha \in P$
- $d[X, i, j]$: list of active items rooted by X with span $[i \dots j]$
- $h[X, i, j]$: list of passive items rooted by X with span $[i \dots j]$
- $d[S, i, j]$: list of active items rooted by S with span $[i \dots j]$
- $h[S, i, j]$: list of passive items rooted by S with span $[i \dots j]$

for all axioms $[Y \rightarrow \bullet\alpha, w]$ **do**
  Insert $[Y \rightarrow \bullet\alpha, w]$ into $a[Y]$
end for
for $l \leftarrow 1, \cdots, |f|$ **do**
  for all $i, j \mid j - i = l$ **do**
    for all items $[Y \rightarrow \alpha_1 \bullet \alpha_2, i, j, w]$ (with α_2 not empty) provable from items in $a[Y]$ and $d[Y, i, j]$ and $h[Y, i, j]$ **do**
      Add $[Y \rightarrow \alpha_1 \bullet \alpha_2, i, j, w]$ to $d[Y, i, j]$
    end for
    for all items $[Y \rightarrow \alpha\bullet, i, j, w]$ provable from items in in $a[Y]$ and $d[Y, i, j]$ and $h[Y, i, j]$ **do**
      Add $[Y \rightarrow \alpha\bullet, i, j, w]$ to $h[Y, i, j]$
    end for
  end for
end for

4.2.2 Translation Generation

Translation generation can be incorporated in the CYK+ parser by adding a component to the items of the deductive proof system described above (Section 4.2.1). In the same fashion as for rule weights, each axiom carries the target side γ of the rule from which it was created. Equation 4.8 shows the axioms created for each hierarchical rule, where γ is simply the target side of the rule from which the axiom is created.

$$\frac{}{[X \rightarrow \bullet\alpha, w, \gamma]} \quad (X \xrightarrow{w} \langle\alpha, \gamma\rangle \in P) \tag{4.8}$$

Similarly to rule weights, all items created from these axioms carry a partial translation that is either the target side γ of the axiom or the product $\otimes$ of two target sides γ_1 and γ_2. The items created using lexical rules, shown in Equation 4.9, carry the same partial translation as their antecedent. As seen in Section 4.2.1, these rules simply create new items by consuming terminal symbols in the source side α of their antecedent.

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w, \gamma]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w, \gamma]} \tag{4.9}$$

Non-lexical rules are shown in Equation 4.10. As seen above (Section 4.2.1), these rules process a non-terminal symbol X in the source side α_1 of an active antecedent $[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_1]$ using a passive antecedent $[X \rightarrow \beta \bullet j, k, w_2, \gamma_2]$. On the target language side of these items, a non-terminal X in γ_1 of the first antecedent is replaced by γ_2. This operation is denoted by $\otimes$. Equation 4.10 shows the non-lexical inference rule with integrated partial translations.

$$\frac{[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_1] \quad [X \rightarrow \beta\bullet, j, k, w_2, \gamma_2]}{[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2, \gamma_1 \otimes \gamma_2]} \tag{4.10}$$

While all parse items carry a partial translation γ, only passive items have con-

sumed the input side of a rule corresponding to γ. We follow the terminology of [Koehn et al., 2007] and denote the partial translations in passive parse items by *translation options*. Translation options are used to generate the translations corresponding to the parse tree resulting from CYK+ parsing.

When searching for the 1-best parse using CYK+ (Section 4.2.1) equivalent items in the item lists are merged into the ones with highest weight. This notion of equivalence does not consider the target sides γ_i of parse items. So the following items are considered as equivalent:

$$[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_1] \Leftrightarrow [X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_2, \gamma_2] \tag{4.11}$$

Their merge is the item with highest weight.

4.2.3 Example

We illustrate CYK+ parsing with the hierarchical grammar given in Figure 4.6 as well as the input sentence F:

F *l'ingénérie informatique est pratique*

From the hierarchical rules, the axioms in Figures 4.7 and 4.8 are created following Equation 4.8. According to Algorithm 1, these are put into the lists $a[X]$ and $a[S]$:

$r_1 : X \xrightarrow{0.1} \langle$ pratique, process $\rangle$	$r_2 : X \xrightarrow{0.4} \langle$ pratique, practice $\rangle$
$r_3 : X \xrightarrow{0.5} \langle$ pratique, practical $\rangle$	$r_4 : X \xrightarrow{1.0} \langle$ l' ingénérie, engineering $\rangle$
$r_5 : X \xrightarrow{0.6} \langle$ informatique, software $\rangle$	$r_6 : X \xrightarrow{0.4} \langle$ informatique, computer science $\rangle$
$r_7 : X \xrightarrow{1.0} \langle$ l' ingénérie X_1, X_1 engineering $\rangle$	$r_8 : X \xrightarrow{1.0} \langle$ X_1 est X_2, X_1 is X_2 $\rangle$
$r_9 : S \xrightarrow{1.0} \langle$ X_1, X_1 $\rangle$	$r_{10} : S \xrightarrow{1.0} \langle$ S X, S X $\rangle$

Figure 4.6: Example hierarchical grammar

Using the inference rule for terminal symbols (Equation 4.9) on the input sentence, a first series of items can be inferred from the axioms. For

$a_1 : \overline{_{X\rightarrow\bullet}\text{ pratique}, 0.1, \text{ process}}$	$a_2 : \overline{_{X\rightarrow\bullet}\text{ pratique}, 0.4, \text{ practice}}$
$a_3 : \overline{_{X\rightarrow\bullet}\text{ pratique}, 0.5, \text{ practical}}$	$a_4 : \overline{_{X\rightarrow\bullet}\text{l' ingénérie}, 1.0, \text{ engineering}}$
$a_5 : \overline{_{X\rightarrow\bullet}\text{informatique}, 0.6, \text{ software}}$	$a_6 : \overline{_{X\rightarrow\bullet}\text{informatique}, 0.4, \text{ computer science}}$
$a_7 : \overline{_{X\rightarrow\bullet}\text{l' ingénérie}X_1, 1.0, X_1 \text{ engineering}}$	$a_8 : \overline{_{X\rightarrow\bullet}\ X_1 \text{ est } X_2, 1.0, X_1 \text{ is } X_2}$

Figure 4.7: Axioms created from rules having lhs X put into $a[X]$.

$a_9 : \overline{_{S\rightarrow\bullet}X_1, 1.0, X_1}$	$a_{10} : \overline{_{S\rightarrow\bullet}\ S\ X, 1.0, S\ X}$

Figure 4.8: Axioms created from rules having lhs S put into $a[S]$.

instance, the passive item $[X \rightarrow$ *informatique*$\bullet, 2, 3, 0.6,$ *software*$]$ can be inferred from the axiom $[X \rightarrow \bullet$*informatique*$, 0.6,$ *software* $]$. In the same fashion, the active item $[X \rightarrow$ *l'* $\bullet$ *ingénérie*$, 0, 1, 1.0,$ *engineering*$]$ can be inferred from $[X \rightarrow \bullet$ *l' ingénérie*$, 1.0,$ *engineering*$]$. As seen in Section 4.2.2 the partial translation of the inferred items is the same as for the antecedents. Further items are created using the inference rules for non-lexical items given in Equation 4.10. For spans of length 3 in the input sentence, the passive item $[X \rightarrow$ *l' ingénérie* $X_1\bullet, 0, 3, 0.6,$ *software engineering* $]$ can be inferred from $[X \rightarrow$ *l' ingénérie* $\bullet$ $X_1, 0, 2, 1.0, X_1$ *engineering* $]$ and $[X \rightarrow$ *informatique*$\bullet, 2, 3, 0.6,$ *software*$]$. As seen in Section 4.2.2, the partial translation of this item is obtained by replacing the non-terminal X_1 in the partial translation "X_1 *engineering*" (in the first antecedent) by "*software*" (in the second antecedent).

Figures 4.9 to 4.11 show some[8] items inferred from the axioms in Figures 4.7 and 4.8 on increasing spans of the sentence F. In addition to the item elements indicated in Equations 4.9 and 4.10, each item carries an identity integer and a list of back-pointers to the identity of its antecedents. For items without antecedent, the back-pointers are omitted. We do not display the partial translation in active items as these are not used for translation generation. Finally, according to Algorithm 1 the items spanning the entire sentence (Figure 4.11) are of two types. The first are either rooted by X or by S. Except for their lhs, the items are exactly the same so we write $[X/S \rightarrow \alpha_1 \bullet \alpha_2, i, j, w]$ to denote *two* items, the first rooted by X and the

[8]For instance, for the span 0-5 (entire sentence) we only display a subset of items that can be created from previous ones. Similarly, we do not display all items inferred for the span 0-3.

second by S.

$[X \rightarrow$ l' ingénérie • $X_1, 0, 2]$ $[1.0, 13, (9)]$ $[X \rightarrow$ l' ingénérie•$, 0, 2]$ $[1.0,$ engineering$, 12, (8)]$		$[X \rightarrow X_1$ est• $X_2, 2, 4]$ $[0.6, 11, (7)]$ $[X \rightarrow X_1$ est• $X_2, 2, 4]$ $[0.4, 10, (6)]$		
$[X \rightarrow$ l' • ingénérie $X_1, 0, 1]$ $[1.0, 9]$ $[X \rightarrow$ l' • ingénérie$, 0, 1]$ $[1.0, 8]$		$[X \rightarrow X_1$ • est $X_2, 2, 3]$ $[0.6, 7, (2)]$ $[X \rightarrow X_1$ • est $X_2, 2, 3]$ $[0.4, 6, (1)]$ $[X \rightarrow$ informatique•$, 2, 3]$ $[0.6,$ software$, 2]$ $[X \rightarrow$ informatique•$, 2, 3]$ $[0.4,$ computer science$, 1]$		$[X \rightarrow$ pratique•$, 4, 5]$ $[0.1,$ process$, 5]$ $[X \rightarrow$ pratique•$, 4, 5]$ $[0.5,$ practical$, 4]$ $[X \rightarrow$ pratique•$, 4, 5]$ $[0.4,$ practice$, 3]$
l'	ingénérie	informatique	est	pratique
0 1	1 2	2 3	3 4	4 5

Figure 4.9: Active (in red at top) and passive parse items of spans 1 and 2 created with rules having lhs X.

The CYK+ algorithm presented so far only searches for the 1-best parse of an input sentence. As mentioned in Section 4.2.1, equivalent items in each cell of the item lists are merged into the one with the highest weight. Figure 4.12 displays the items generated by the CYK+ parser when equivalent items are merged. The identity integers remain the same as in the previous figures. When several items have the same weight, we randomly choose one.

4.2.4 N-best list Generation

For applications in statistical machine translation (SMT), it is not sufficient to obtain the 1-best derivation for an input sentence. In order to tune SMT models using Minimum Error Rate Training [Och, 2003], an n-best list of possible translations is required. As partial translations for parse items are only provided in passive parse items (see Section 4.2.2), n-best list generation only applies on these items, which

<table>
<tr><td colspan="4">$[X \rightarrow X_1$ est $\bullet\, X_2, 0, 4]$
$[0.6, 17, (15)]$

$[X \rightarrow X_1$ est $\bullet\, X_2, 0, 4]$
$[0.4, 16, (14)]$</td><td></td></tr>
<tr><td colspan="3">$[X \rightarrow$ l' ingénérie $X_1\bullet, 0, 3]$
$[0.6,$ software engineering$, 15, (13, 2)]$

$[X \rightarrow$ l' ingénérie $X_1\bullet, 0, 3]$
$[0.4,$ computer science engineering$, 14, (13, 1)]$</td><td colspan="2"></td></tr>
<tr><td colspan="2"></td><td colspan="3">$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.06,$ software is process$, 23, (11, 5)]$

$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.04,$ computer science is process$, 22, (10, 5)]$

$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.3,$ software is practical$, 21, (11, 4)]$

$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.2,$ computer science is practical$, 20, (10, 4)]$

$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.24,$ software is practice$, 19, (11, 3)]$

$[X \rightarrow X_1$ est $X_2\bullet, 2, 5]$
$[0.16,$ computer science is practice$, 18, (10, 3)]$</td></tr>
<tr><td colspan="2">l'</td><td>ingénérie</td><td>informatique</td><td>est</td><td>pratique</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>2</td><td>2</td><td>3</td><td>3</td><td>4</td><td>4</td><td>5</td></tr>
</table>

Figure 4.10: Active (in red at top) and passive parse items of span 3 created with rules having lhs X.

we call *translation options*. In [Chiang, 2007], n-best lists are generated following Algorithm 2 below. Its core is the function $GENBEST(l)$, which takes a tuple of ordered lists L_i of translation options as input and generates an ordered list of these. $GENBEST(l)$ is recursively applied each time a translation option is inferred.

To illustrate n-best list generation, suppose that we want to compute the n-best list of translations for the parse items with ids 18 to 23 in the chart in Figure 4.10. We recursively apply $KBest(x)$ to all chart items used to infer 18 to 23, that is 10,

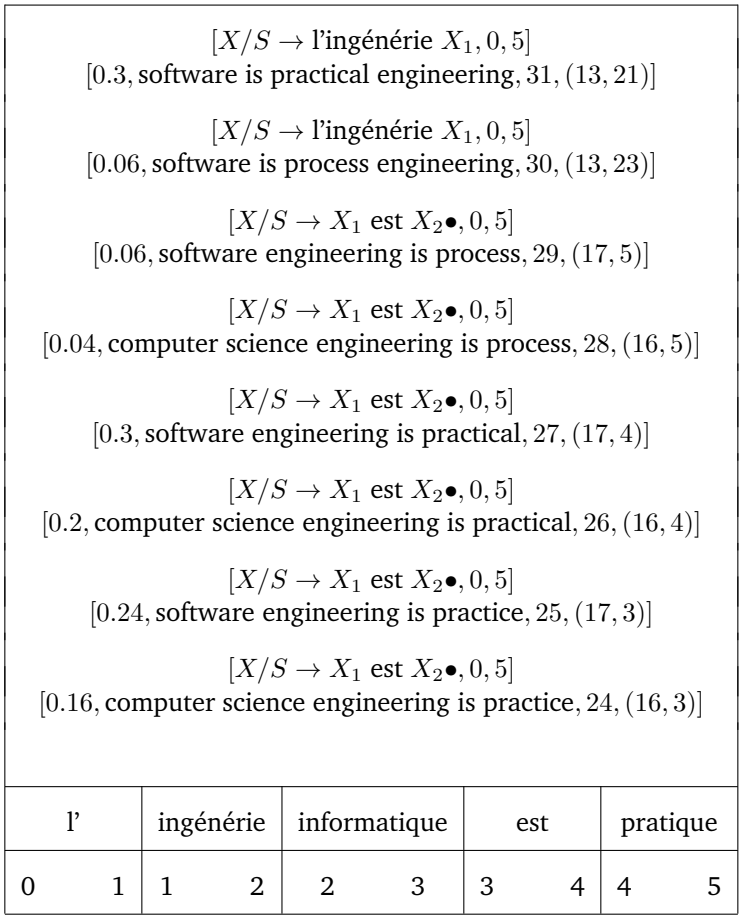


Figure 4.11: Active (in red at top) and passive parse items of span 5 created with rules having lhs X or S.

11, 3, 4, 5 in Figure 4.10. As 3, 4 and 5 are directly inferred from axioms, they are ordered and put together in a list L_1 which is added to ℓ. As 10 and 11 are inferred from the further items 6 and 7, $KBest(x)$ is applied to those items, which in turn are inferred from 1 and 2. The application of $KBest(x)$ to the items 1 and 2 returns an ordered list of these two items, that is $(2, 1)$. As 10 and 11 are directly inferred from 6 and 7 which in turn are directly inferred from 1 and 2, the latter are put in a second (ordered) list L_2 which is also added to ℓ. The function $GENBEST(\ell)$ is then applied to L_1 and L_2 and returns an ordered list of items 18 to 23, that is $(21, 19, 20, 18, 23, 22)$.

One way to define $GENBEST(\ell)$ is to make it compute all elements of the

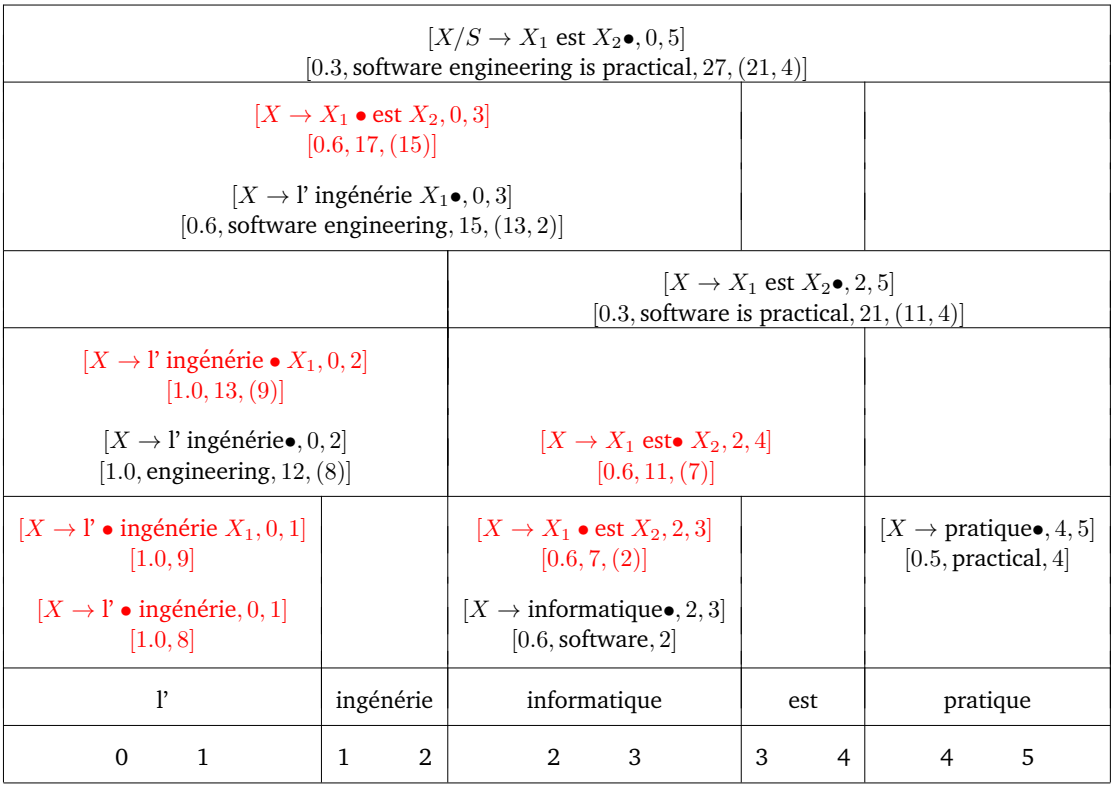


Figure 4.12: Chart item lists with merged equivalent items.

product of the input lists, which generates a very large list of products although in practice, only the top n elements of such a list are needed. So there is no need to compute all products. In order to generate such n-best lists, [Chiang, 2007] uses an algorithm that lazily computes the product of several ordered input lists. The main idea of this procedure is that when the input lists are ordered[9], some product values are guaranteed to be smaller than others. For instance, the product between the first element of all lists is always smaller than the product of all other elements in the list. In the same fashion, the product of elements that are adjacent to the best product are guaranteed to be smaller than all other products.

[9]We assume that ordered lists range from smaller to larger elements

Algorithm 2 N-best list generation

Data structures:
- $best[t]$:ordered list of best translation options
- ℓ:tuple of translation option lists

1: **Function** $KBest(v)$
2: **for** $(u_1, ..., u_n)$ such that v inferred from $(u_1, ..., u_n)$ **do**
3: Add $\langle KBest(u_1) \cdots KBest(u_n) \rangle$ to ℓ
4: **end for**
5: **return** $GENBEST(\ell)$

Lazy computation of sorted list products

The product computation procedure is given in Algorithm 4.2.4. It uses four data structures:

- A tuple of virtual lists, denoted by ℓ.
- A priority queue containing candidate values for products between lists, denoted by $prods[]$.
- A tuple $\langle \otimes_i L_i(k), L, r \rangle$ containing the result of the product applied to the i input lists L_i, an index L over the lists and an i-dimensional list index r.

The algorithm begins by computing the product $\otimes_i L_i(k)$ for the first element of all lists and inserting it into $prods[]$ (lines 2-5), its index, stored in the variable r, is **1**, that is a sequence of 1. Then, the best element of $prods[]$ is retrieved and stored in $bestProds[]$ (lines 18-20). As long as the size of $bestProds[]$ is smaller than the desired size n, the procedure continues by (i) determining the indices r'_i of adjacent elements to the one at index r (by adding the bolean vector b^i to r) (line 10) and (ii) computing the product of these elements which is then added to $prods[]$ (lines 11-15) and (iii) retrieving the best element of $prods[]$ which is stored in $bestProds[]$ (lines 18-20).

We illustrate the functioning of Algorithm 4.2.4 with two lists L_1 and L_2 containing structures of two elements: (i) a partial translation and (ii) a probability score:

Algorithm 3 Lazy computation of sorted list products

Data structures:
- ℓ:tuple of virtual lists
- $prods[]$:priority queue containing candidate products
- $\langle \otimes_i L_i(k), \mathbf{L}, r \rangle$:product of list cells at i-dimensional index r
- $bestProds[]$:ordered list of n-best products

1: **FUNCTION** $MERGEPROD(\ell, \otimes)$
2: **for** $\mathbf{L} \in \ell$ **do**
3: INSERT ($prods[], \langle \otimes_i L_i(k), \mathbf{L}, \mathbf{1} \rangle$)
4: HEAPIFY($prods[]$)
5: **end for**
6: $\mathbf{L} \leftarrow nil$, $\mathbf{r} \leftarrow nil$
7: **while** $\mid bestProds[] \mid < n$ **do**
8: **if** $\mathbf{L} \neq nil$ **then**
9: **for** $i \leftarrow 1, \cdots, |L|$ **do**
10: $r' \leftarrow r + b^i$
11: **if** $L_i(r')_i$ defined **then**
12: **if** $\langle \otimes_i L_i(r'), \mathbf{L}, r' \rangle \notin prods[]$ **then**
13: INSERT($prods[], \langle \otimes_i L_i(r'), \mathbf{L}, r' \rangle$
14: **end if**
15: **end if**
16: **end for**
17: **end if**
18: **if** $prods[] > 0$ **then**
19: $\langle x, \mathbf{L}, r \rangle \leftarrow EXTRACTBEST(prods[])$
20: insert x into $bestProds[]$
21: **else**
22: **return** undefined
23: **end if**
24: **end while**
25: **return** $bestProds[]$

L_1:

1 ⟨software,0.6⟩

2 ⟨computer science,0.4⟩

L_2:

1 ⟨X is practical,0.5⟩

2 ⟨X is practice,0.4⟩

3 ⟨X is process,0.1⟩

We define the multiplication operator $\otimes$ in the same way as in Section 4.2.2. The elements of L_1 and L_2 used for product computation can be illustrated using a 2-dimensional grid such as the one in Figure 4.13. Suppose that we want to generate the 3-best products of these two lists using Algorithm 4.2.4. We begin by computing the product of the first elements of the lists (lines 2-5) and add it to $prods[]$. Then, we extract ⟨software is practical,0.3⟩ from $prods[]$ and add it to $bestProds[]$ (lines 18-20, blue item (1) in Figure 4.13). At this point, $bestProds[]$ has size 1 so we perform a second iteration and compute the indices of the elements that are adjacent to the one at index $(1,1)$, that is $(2,1)$ and $(1,2))$ (line 10) before calculating the product of the corresponding elements in the lists, which we add to $prods[]$ (lines 11-15, red items (2) in Figure 4.13). At this point, $prods[]$ has size 2 so we perform a second iteration and extract ⟨ software is practice,0.24⟩ from $prods[]$ to put it into $bestProds[]$ (lines 19-20). Then, we compute the indices of the elements that are adjacent to the one at index $(2,1)$, that is $(3,1)$ and $(2,2)$ (line 10) and calculate the product of the corresponding elements in the lists, which we add to $prods[]$ (lines 11-15, magenta items (3) in Figure 4.13). Finally we extract ⟨ computer science is practical,0.2⟩ from $prods[]$ and add it to $bestProds[]$ which now has the desired size 3.

L1/L2	⟨software,0.6⟩	⟨computer science,0.4⟩
⟨X is practical,0.5⟩	(1) ⟨ software is practical,0.3⟩	(2) ⟨ computer science is practical,0.2⟩
⟨X is practice,0.4⟩	(2) ⟨ software is practice,0.24⟩	(3) ⟨ computer science is practice,0.16⟩
⟨X is process,0.1⟩	(3) ⟨ software is process,0.06⟩	

Figure 4.13: Example of lazy product computation

4.2.5 Language Model Integration

Besides rule specific features, the hierarchical translation model (Section 4.1.2) consists of a language model feature. When computing n-best lists, this feature must be integrated into the translation options. As seen in Section 4.2.2, translation options carry, in addition to their weight, a translation γ of their source side. In order to define a language model score over these items, two functions are added to γ ([Chiang, 2007]):

- $P_{LM}(\gamma)$ computes the m-gram language model score for a segment of $\gamma = a_1 \dots a_{|\gamma|}$ of terminals, ignoring words marked with $*$:

$$P_{LM}(\gamma) = \prod_{\substack{m<i<|\gamma| \\ *\notin a_{i-m+1} \dots a_{i-1}}} p(a_i \mid a_{i-m+1} \cdots a_{i-1}) \tag{4.12}$$

- $Mark_m$ marks words for which the language model has already been computed in segments bigger than m:

$$Mark_m = \begin{cases} a_1 \cdots a_{m-1} * a_{|\gamma|-m+2} \cdots a_{|\gamma|} \text{ if } \mid \gamma \mid \geq m \\ a_1 \cdots a_{|\gamma|} \text{ otherwise} \end{cases} \tag{4.13}$$

To illustrate the functioning of functions P_{LM} and $Mark_m$, we compute the 3-gram language model score of the segment $\gamma =$ *computer (c) science (s) engineering (e) is (i) practical (p)*. The language model score is $P_{LM}(\gamma) = p(e|cs)\dot{p}(i|se)\dot{p}(p|ei)$. The function $Mark_m$ marks the third word of gamma, i.e. $Mark_3(\gamma) = cs * ip$.

For chart parsing and n-best list generation, the functions P_{LM} and $Mark_m$ are applied on the partial translations γ. The resulting parse items are shown in Equations 4.14 and 4.15:

$$\frac{[X \to \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w, \gamma, Mark_m(\gamma), P_{LM}(\gamma)]}{[X \to \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w, \gamma, Mark_m(\gamma), P_{LM}(\gamma)]} \tag{4.14}$$

$$\frac{[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_1, Mark_m(\gamma_1), P_{LM}(\gamma_1)] \quad [X \rightarrow \alpha_2\bullet, j, k, w_2, \gamma_2, Mark_m(\gamma_2), P_{LM}(\gamma_2)]}{[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2, Mark_m(\gamma_1 \otimes \gamma_2), P_{LM}(\gamma_1 \otimes \gamma_2)]} \tag{4.15}$$

Following the terminology of [Koehn et al., 2007], we call translation options that integrate language model scoring *hypotheses*. Algorithm 1 can easily be modified to integrate language model scoring by replacing the translation options in $h[X]$ and $h[S]$ by the items of the form given in Equations 4.14 and 4.15. Note that the correctness of the n-best algorithm is lost by integrating the language model.

4.2.6 Pruning

The CYK+ chart parsing algorithm for hierarchical grammars generates a large number of items in each cell of the chart. When language model scores are integrated into the parsing procedure, time and memory consumption of decoding become prohibitive. In order to reduce these costs, [Chiang, 2007] removes items with log score worse than:

(1) β added to the best score in the same cell

(2) the score of the b-th best item in the cell

When language model computation is added to translation options, those are not directly comparable because the probability of generating the first $(m-1)$ words of γ is not computed. For pruning, [Chiang, 2007] computes the language model score of the $m-1$ first words of γ for each cell and adds it to the score of the cell.

One way to prune the chart cells is to compute all items for each cell and then remove the ones that meet the pruning conditions. But inferring items that are to be removed later is suboptimal and [Chiang, 2007] defines a pruning procedure, called cube pruning, that only computes chart items that will not be pruned away later on.

(i_1) [informatique est •X,2,5,0.6,software is X]
(i_2) [informatique est •X,2,5,0.4,computer science is X]

(j_1) [pratique•,5,6,0.5,practical]
(j_2) [pratique•,5,6,0.4,practice]
(j_3) [pratique•,5,6,0.1,process]

Figure 4.14: Example Input lists to Cube Pruning

Cube Pruning

The main idea of cube pruning is the lazy computation of chart items in the CYK+ algorithm with integrated language model. In this procedure, chart items that are combined to infer translation options are modeled as (ordered) n-best lists. Computation of inferred items is then done using the function $MERGEPROD(\ell, \otimes)$ given in Algorithm 4.2.4. The integration of cube pruning in the CYK+ chart is simply a matter of replacing line 9 of Algorithm 1 with the subroutine below. The pruning conditions refer to conditions (1) and (2) in Section 4.2.6.

Algorithm 4 Cube pruning of chart cells

1: **for all** items $[X \rightarrow \alpha\bullet, i, j, w]$ provable from items in $a[X]$ and $d[X]$ and $h[X]$ **do**
2: $L_{ax} \leftarrow Sort(a[X])$ and $L_{dx} \leftarrow Sort(d[X])$ and $L_{hx} \leftarrow Sort(h[X])$
3: Add L_{ax}, L_{dx} and L_{hx} to ℓ
4: **while** $h[X][i]$ does not meet pruning conditions **do**
5: $h[X] \leftarrow MERGEPROD(\ell, \otimes)$
6: **end while**
7: **end for**

To illustrate cube pruning, consider the item lists $I = (i_1, i_2)$ and $J = (j_1, j_2, j_3)$, given in Figure 4.14. They can be used to infer several hypotheses $h(i, j)$ that go to the same chart cell (spanning from 2 to 6). Instead of computing all possible hypotheses for this cell, cube pruning lazily computes the hypotheses, as shown in Figure 4.15. The 3-best hypotheses $h(1, 1)$, $h(1, 2)$ and $h(2, 1)$ are given in Figure 4.16. We assume that the partial translations are scored using a 3-gram language model.[10]

[10]The language model scores are constructed for the example.

I/J	(j1)	(j2)	(j3)
(i1)	h(1,1)	h(1,2)	
(i2)	h(2,1)		

Figure 4.15: Cube Pruning Example

1 [informatique est pratique•,2,6,0.3,*software is practical*,*software is * is practical*, 0.25]
2 [informatique est pratique•,2,6,0.24,*software is practice*,*software is * is practice*, 0.1]
3 [informatique est pratique•,2,6,0.2,*computer science is practical*,*computer science * is practical*, 0.15]

Figure 4.16: 3-best hypotheses computed using Cube Pruning

4.3 SMT with syntactically annotated SCFG

As seen in Section 2.2.1, several authors build SMT systems with syntactically annotated SCFG rules. In the next sections, we briefly present several strategies to extract such rules from parallel corpora. The rule extraction strategies for Sh-l-MBOT grammars (Section 5.1) are extensions of the heuristics presented here.

4.3.1 Training Data for SMT with Syntactic annotation

The extraction and training of grammar rules that include syntactic annotation is performed on a word aligned parallel corpus with corresponding parse tree on each side. Such a tree pair is called a *word aligned biparsed sentence pair*. Figure 4.17 shows such a tree pair for English and German sentences. The leaves of each tree are the words of the sentence pair. The non-terminals belong to the parse tree of each sentence. The alignment is between the leaves of the trees. For a given translation direction, e.g. English-to-German, we call the aligned tree of the input language the *source tree* and the tree of the output language the *target tree*.

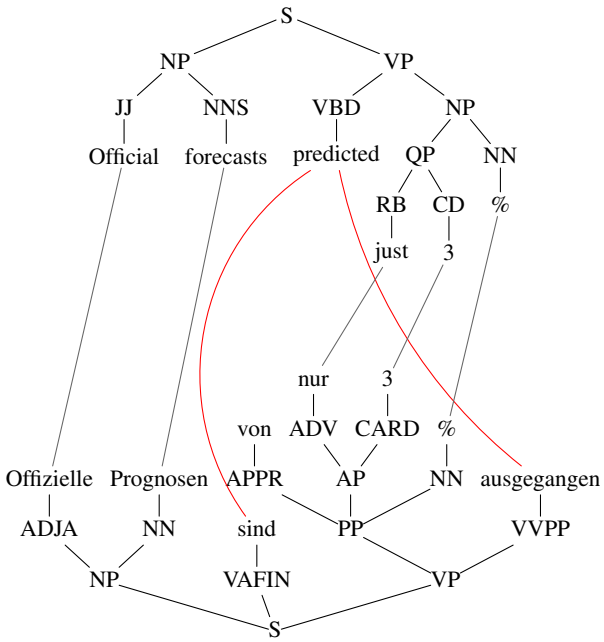


Figure 4.17: Word aligned biparsed sentence pair

4.3.2 SCFG rules as Decorated Hierarchical Rules

The extraction of hierarchical rules decorated with source and target language syntactic annotations follows the two-step heuristic in [Chiang, 2007], presented in Section 4.1.1 except that (i) the extraction procedure is performed on word aligned biparsed sentence pairs and (ii) additional conditions constrain the rule extraction heuristic. This approach is implemented in the syntax component of the Moses open source toolkit [Hoang et al., 2009]. The additional constraints on source and target language sequences f_i^j and $e_{i'}^{j'}$ can be formulated as:

4. The source language phrase f_i^j must match a syntactic constituent in the source parse tree.

5. The target language phrase $e_{i'}^{j'}$ must match a syntactic constituent in the target parse tree.

The left-hand side of initial rules[11] is a pair of non-terminals, namely those matched in conditions 4 and 5.

The second step of the heuristic is the same as in Section 4.1.1: new rules are created by excising already collected rules from larger ones. This process leads to a rule containing gaps on the source and target rhs. These are replaced by the (annotated) non-terminals in the source and target lhs of the excising rules.

To illustrate SCFG rule extraction with decorated non-terminals, we show the word aligned segments in Figure 4.1 (Section 4.1.1) together with their parse trees in Figure 4.18.

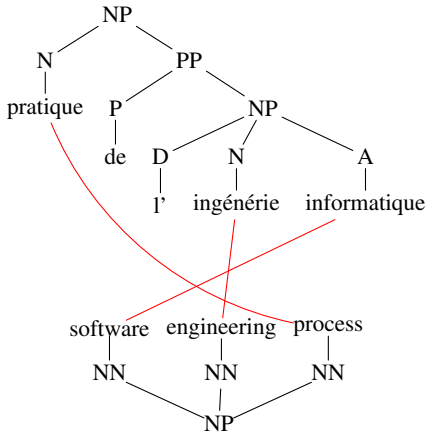


Figure 4.18: Word aligned biparsed sentence pair

The initial rules extracted from this pair are p_1 to p_4 below.[12] Note that the rules r_2, r_3 and r_5 in Section 4.1.1 cannot be extracted anymore because they do not match a syntactic constituent.

p_1 (N,NN) $\rightarrow$ ⟨ pratique, process ⟩

p_2 (N,NN) $\rightarrow$ ⟨ ingénérie, engineering ⟩

p_3 (A,NN) $\rightarrow$ ⟨ informatique, software ⟩

p_4 (NP,NP) $\rightarrow$ ⟨ pratique de l' ingénérie informatique, software engineering process ⟩

[11] By *initial rules*, we denote the hierarchical rules that contain no non-terminals in their right-hand sides.

[12] We use a notation similar to the one in Section 2.1.1 and write SCFG rules as $(A_s, A_t) \rightarrow \langle \alpha, \beta, \widetilde{A} \rangle$, except that we indicate the alignment $\widetilde{A}$ by indexing non-terminal symbols.

In the second step of the rule extraction heuristic, rules p_1 to p_3 are subtracted from p_4 to create rules such as p_5 to p_7.

p_5 (NP,NP) $\rightarrow \langle$ N_1 de l' ingénérie informatique, software engineering NN_1 $\rangle$

p_6 (NP,NP) $\rightarrow \langle$ pratique de l' N_1 informatique, software NN_1 process $\rangle$

p_7 (NP,NP) $\rightarrow \langle$ pratique de l' ingénérie A_2, NN_2 engineering process $\rangle$

Because the annotated rule extraction yields much fewer rules than the hierarchical rule extraction, the additional restrictions in [Chiang, 2007] (Section 4.1.1) can be ignored. As a consequence, rules like p_8 and p_9 can be extracted:

p_8 (NP,NP) $\rightarrow \langle$ pratique de l' N_1 A_2, NN_2 NN_1 process $\rangle$

p_9 (NP,NP) $\rightarrow \langle$ N_1 de l' N_2 A_3, NN_3 NN_2 NN_1 $\rangle$

For more flexibility, glue rules can be used in SCFG grammars with annotated non-terminals. In this setup, the set of glue rules contains all rules that concatenate labeled source and target non-terminals $n \in N_s$ and $n' \in N_t$

(S,S) $\rightarrow \langle$ S_1 n, S_1 n' $\rangle$

(S,S) $\rightarrow \langle$ n, $n'\rangle$

The set of glue rules for syntactically annotated grammars contain all possible rules obtained by annotating the non-terminals in rules g_1 and g_2 in Section 4.1.1 with syntactic labels.

4.3.3 Hierarchical rules with target annotations

Hierarchical rules with syntactic annotation on the target language side only can easily be obtained by removing constraint (4) from the heuristic presented in the previous section. Consequently, the non-terminal for the source side is X as in hierarchical grammars. The input to this heuristic are word aligned parallel sentences where only the target sentence carries a parse tree. An example is given in Figure 4.19.

Rules q_1 to q_9 below show example hierarchical rules with target side syntactic annotations extracted from the sentence pair in Figure 4.19. Rules q_1 to q_5 are ex-

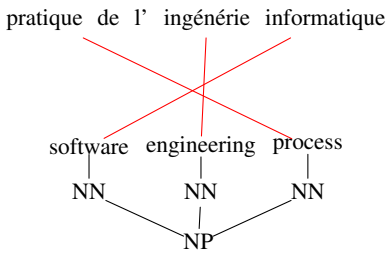


Figure 4.19: Word aligned sentence pair with parse tree on target language side.

tracted according to the first step of the heuristic presented in the previous section. Rules q_6 to q_9 are extracted according to the second step. These example rules show that the removal of the source syntactic constraint (4) allows to extract additional rules such as q_3 and q_8

q_1 (X,NN) $\rightarrow$ ⟨ pratique, process ⟩
q_2 (X,NN) $\rightarrow$ ⟨ ingénérie, engineering ⟩
q_3 (X,NN) $\rightarrow$ ⟨ de l'ingénérie, engineering ⟩
q_4 (X,NN) $\rightarrow$ ⟨ informatique, software ⟩
q_5 (X,NP) $\rightarrow$ ⟨ pratique de l' ingénérie informatique, software engineering process ⟩
q_6 (X,NP) $\rightarrow$ ⟨ X_1 de l' ingénérie informatique, software engineering NN_1 ⟩
q_7 (X,NP) $\rightarrow$ ⟨ pratique de l' X_1 informatique, software NN_1 process ⟩
q_8 (X,NP) $\rightarrow$ ⟨ pratique X_1 informatique, software NN_1 process ⟩
q_9 (X,NP) $\rightarrow$ ⟨ X_1 de l' ingénérie X_2, NN_2 engineering NN_1 ⟩

The set of glue rules consists of all possible rules obtained by annotating the target non-terminals X in rules g_1 and g_2 (Section 4.1.1) with syntactic labels.

4.3.4 SCFG Rules as Shallow STSG Rules

Shallow Synchronous Tree Substitution Grammars (sh-STSG) are a restricted variant of the STSG grammars presented in Chapter 2 (Section 2.1.2). The restriction on STSG rules to obtain their shallow variant is that the left-hand-side (LHS) has height at most 2.

Automatic extraction of Shallow Minimal STSG Rules

Shallow STSG rules are obtained by first extracting minimal STSG rules following the procedure in [Liu et al., 2009] and then making those shallow by removing their internal nodes.

Extraction of Minimal STSG Rules Given a word aligned biparsed parallel corpus such as in Figure 4.17, the extraction procedure for minimal STSG rules can be summarized as shown in Figure 4.20. The detailed steps are presented in Figures 4.21 to 4.24. Figure 4.25 displays all rules extracted after one iteration of the STSG rule extraction procedure as well as the remaining word aligned tree pair. All words could be extracted to form rules except *predicted* and *sind ausgegangen* because the maximal source node consistent with the alignment to the leaves *sind* and *ausgegangen* is the root node of the word aligned tree pair. Extracting a rule that contains this node requires to extract first all rules arising from tree pairs that are lower in the source tree. Figures 4.26 and 4.27 show the rules extracted after a second and third pass of the rule extraction algorithm. Finally, Figure 4.28 shows the last step of the procedure, where the remaining tree pair is the rule containing *predicted* and *sind ausgegangen*.

Step 1	Find a maximal source node consistent with a (minimal) set S of word alignment links
Step 2	Find a maximal target node consistent with the minimal extension of S *Reiterate 1 and 2 if necessary*
Step 3	Excise the rule with maximal source and target node from the biparsed sentence pair and replace with parent labels
Step 4	Add the rule to the rule set
Step 5	Repeat

Figure 4.20: Main steps of the rule extraction procedure for minimal STSG rules.

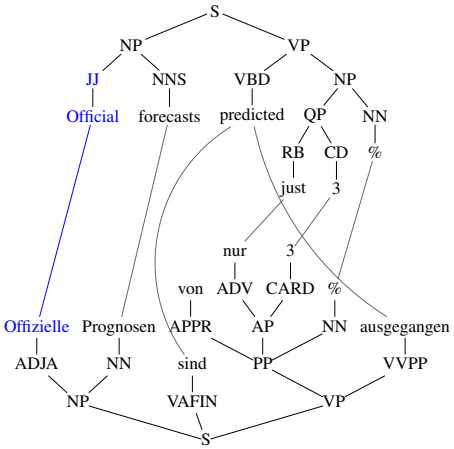
Step 1	**Find maximal source node consistent with word alignment links**
Algorithm	Given a lowest node l_s in the source tree that is not a non-terminal leaf and that is aligned to a set of leaves $\{l_{t1}, ..., l_{tn}\}$ in the target tree, find the highest node in the source tree such that no leaf in its subtree aligns to $l_{tk} \notin \{l_{t1}, ..., l_{tn}\}$.
Example	We assume that the English tree is the source tree. The source leaf node $l_s = Official$ is aligned to the target leaf node $l_{t1} = Offizielle$. The maximal node in the source tree that still aligns to only l_{t1} is labeled JJ. 

Figure 4.21: First step of STSG rule extraction procedure.

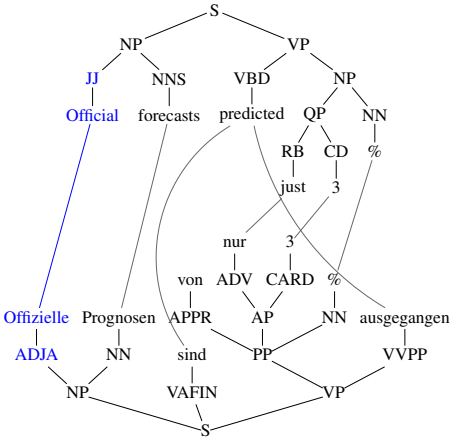
Step 2	**Find maximal target node consistent with word alignment links**
Algorithm	Given a set of target leaves $l_{ti} \in \{l_{t1}, ..., l_{tn}\}$ in the target tree that is aligned to the source leaves $\{l_{s1}, ..., l_{sm}\}$, find the highest node in the target tree such that it contains all leaves $l_{ti} \in \{l_{t1}, ..., l_{tn}\}$.
Example	We assume that the English tree is the source tree. The target leaf node is $l_{t1} = Offizielle$. The source leaf node is $l_s = Official$. The maximal node in the target tree that still aligns to l_s is labeled $ADJA$. 

Figure 4.22: Second step of STSG rule extraction procedure.

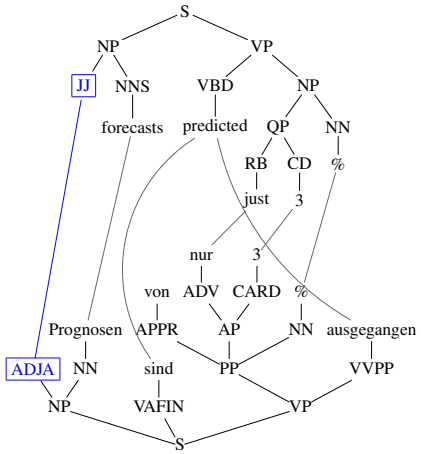
Step 3	**Excise the rule with maximal source and target node**
	Remove the maximal node in the source tree as well as its subtree. Replace the removed subtree by its root. Remove the maximal node in the target tree as well as its subtree. Replace the removed subtree by its root. Align the remaining non-terminals.
Example	Remove the source subtree rooted by JJ and replace with JJ. Remove the target subtree rooted by $ADJA$ and replace with $ADJA$. Align JJ and $ADJA$. 

Figure 4.23: Third step of STSG rule extraction procedure.

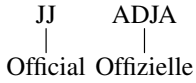
Step 4	**Add rule to rule set**
	Create an STSG rule with the excised trees on the input and output sides.
Example	

Figure 4.24: Fourth step of STSG rule extraction procedure.

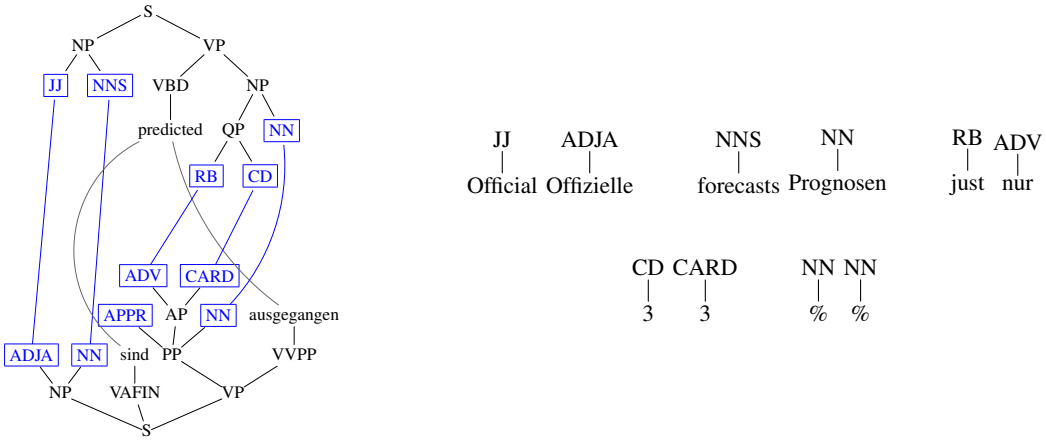


Figure 4.25: Word aligned biparsed sentence pair and extracted rules after the first pass of the algorithm.

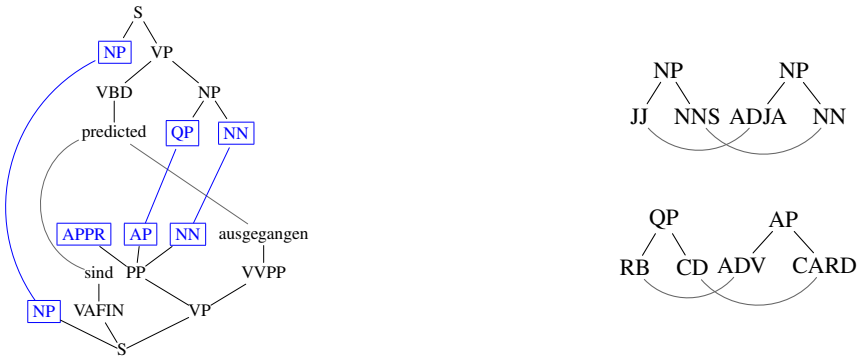


Figure 4.26: Word aligned biparsed sentence pair and extracted rules after the second pass of the algorithm.

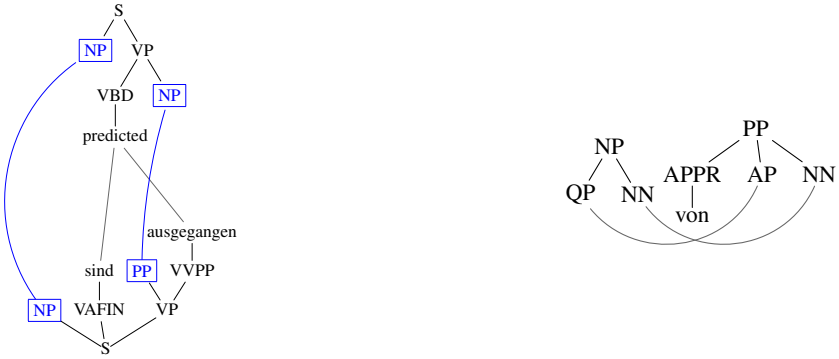


Figure 4.27: Word aligned biparsed sentence pair and extracted rules after the third pass of the algorithm.

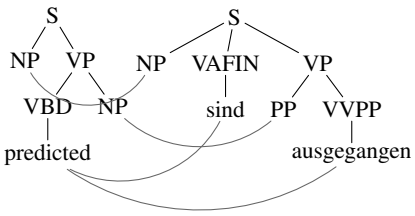


Figure 4.28: Rules extracted after last pass of STSG rule extraction algorithm.

Removal of Internal Nodes All STSG rules extracted above (Paragraph 4.3.4) are shallow except the last one, shown in Figure 4.28. The shallow version of this rule is given in Figure 4.29.

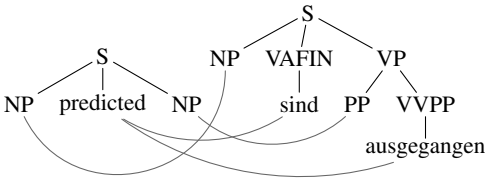


Figure 4.29: STSG rule after removal of internal nodes.

4.3.5 Shallow STSG Rules with Target Annotations

In the same fashion as shallow STSG rules with annotations on both sides, rules with annotations on the target language (string-to-tree) side only can be obtained by extracting STSG string-to-tree rules with the algorithm in [Galley et al., 2004, Galley et al., 2006]. As seen in Section 2.2.1, this method is implemented in the Moses open-source toolkit [Williams and Koehn, 2012]. The rule extraction procedure works in the same way as the minimal rule extraction for STSG rules presented in Section 4.3.4 except that there are no annotations on the source side. Figure 4.30 gives a summary of the extraction algorithm for string-to-tree STSG rules. We also display the shallow version of the string-to-tree STSG rules extracted from the training data in Figure 4.31 with this algorithm.

Step 1	Find the largest string consistent with the word alignment
Step 2	Find a maximal target node consistent with the word alignment *Reiterate 1 and 2 if necessary*
Step 3	Excise the rule with largest string and target node from the biparsed sentence pair and replace the source string with X and the target tree with parent labels
Step 4	Add the rule to the rule set
Step 5	Repeat

Figure 4.30: Main steps of the rule extraction procedure for minimal STSG rules.

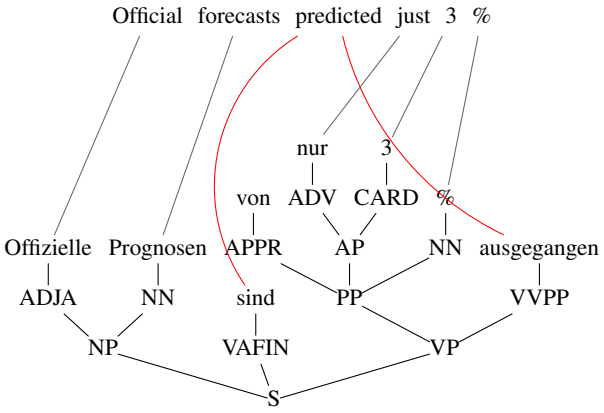


Figure 4.31: Word aligned sentence pair with parse tree on target language side.

s_1 (X,ADJA) $\rightarrow$ ⟨ Official, Offizielle ⟩

s_2 (X,NN) $\rightarrow$ ⟨ forecasts, Prognosen ⟩

s_3 (X,ADV) $\rightarrow$ ⟨ just, nur ⟩

s_4 (X,CARD) $\rightarrow$ ⟨ 3, 3 ⟩

s_5 (X,NN) $\rightarrow$ ⟨ %, %⟩

s_6 (X,NP) $\rightarrow$ ⟨ X_1 X_2, $ADJA_1$ NN_2 ⟩

s_7 (X,AP) $\rightarrow$ ⟨ X_1 X_2, ADV_1 $CARD_2$ ⟩

s_8 (X,PP) $\rightarrow$ ⟨ X_1 X_2, von AP_1 NN_2 ⟩

s_9 (S,X) $\rightarrow$ ⟨ X_1 predicted X_2, NP_1 sind PP_2 ausgegangen ⟩

In contrast to hierarchical rules with target side annotations, shallow string-to-tree STSG rules can contain many adjacent non-terminals on the source side of the rules, which makes decoding very expensive. To limit

the decoding complexity, [Williams and Koehn, 2012] apply scope-3-pruning [Hopkins and Langmead, 2010] to the obtained shallow STSG grammar.

4.4 SCFG Decoding with Syntactic Annotations

4.4.1 CYK+ Chart Parser with Syntactic Annotations

In a similar way to rule extraction, decoding with a syntactically annotated SCFG can be obtained by augmenting the items in a hierarchical decoder (Section 4.2.1) with source and target syntactic labels [Hoang, 2011]. Because non-terminals are labeled, the inference process of the annotated SCFG decoder is more constrained than in the hierarchical case. For instance, non-terminals in the source lhs of the dotted rules must match the input parse tree. In such a setup, the axioms of the chart parser have the form below. We use the notation in [Hoang, 2011] which is based on [Satta and Peserico, 2005].

$$\frac{}{[\bullet A \rightarrow \bullet\alpha, B \rightarrow \beta, w]} \quad (A, B) \xrightarrow{w} \langle \alpha, \beta \rangle \in P \tag{4.16}$$

Where A and B are the labeled source and target left-hand-sides (lhs) of the rule and α and β the source and target right-hand sides (rhs). The dots indicate that both, the source lhs A and the source string α remain to be recognized for the subtree rooted by A to be processed. Thus, passive parse items have the form in Equation 4.17, where the non-terminal A is recognized if it matches the input parse tree.

$$[A\bullet \rightarrow \alpha\bullet, B \rightarrow \beta, i, j, w] \tag{4.17}$$

Processing of non-terminals in the source lhs of dotted rules is given in the following inference rule:

$$\frac{[\bullet A \rightarrow \alpha_1 \bullet, B \rightarrow \beta, i, j, w]}{[A \bullet \rightarrow \alpha_1 \bullet, B \rightarrow \beta, i, j, w]} \quad A \in V_{i,j} \tag{4.18}$$

where $V_{i,j}$ denotes parse labels of the input sentence spanning from i to j.

Lexical inference rules, given in Equation 4.19, work in the same way as in the hierarchical case and simply consume a terminal symbol t_{j+1} spanning from j to $j+1$.

$$\frac{[\bullet A \rightarrow \alpha_1 \bullet t_{j+1} \alpha_2, B \rightarrow \beta, i, j, w]}{[\bullet A \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, B \rightarrow \beta, i, j+1, w]} \tag{4.19}$$

In non-lexical inference rules, the consumed non-terminals and their corresponding target sides are labeled. Hence, when a passive parse item is used to recognize a non-terminal symbol, its source and corresponding target lhs must match the consumed non-terminal as well as the target non-terminal to which it is aligned. The non-lexical inference rule for annotated SCFG decoding is given in Equation 4.20. The source lhs C and target lhs D of the passive item match the aligned source and target non-terminals C_n and D_n that are being processed.

$$\frac{[\bullet A \rightarrow \alpha_1 \bullet C_n \alpha_2, B \rightarrow \beta_1 D_n \beta_2, i, j, w_1] \quad [C \bullet \rightarrow \gamma \bullet, D \rightarrow \delta, j, k, w_2]}{[\bullet A \rightarrow \alpha_1 C_n \bullet \alpha_2, B \rightarrow \beta_1 \delta \beta_2, i, k, w_1 * w_2]} \tag{4.20}$$

The goal of the system is the item $[S \bullet \rightarrow \alpha \bullet, S \rightarrow \beta, 0, |s|, w]$, where the rhs of a rule starting with the start non-terminal (S, S) and spanning the entire input string s has been processed.

4.4.2 Search Procedure and Translation Generation

The search procedure of the CYK+ chart parser for annotated SCFG works much in the same way as the search procedure for the hierarchical decoder (Algorithm 1). The main differences are that (i) the used items are those given in Equations

4.16 and 4.18 to 4.20 and (ii) source lhs non-terminal symbols must be consumed in order to create passive parse items. When the complete source rhs of an item has been consumed, the item is lost if its source lhs cannot be consumed, i.e. if the root of the processed subtree does not match the input parse tree. Note that the dotted chart parser cannot handle unary rules correctly and hence (potentially) cannot parse input trees such as the one in Figure 4.32 correctly. This problem can be alleviated by removing unary nodes in the input parse trees, as done in Figure 4.33.

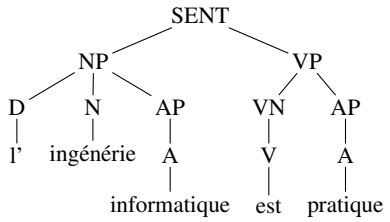


Figure 4.32: Parse tree of French sentence F

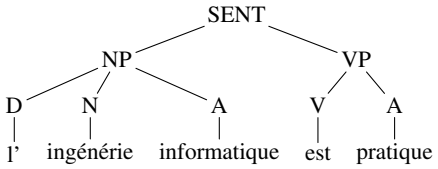


Figure 4.33: Parse tree of French sentence F after removal of unary nodes (except POS tags).

Translation generation works exactly as in the hierarchical case (Section 4.2.2): the partial translation carried by translation options is the target rhs β. The operation $\otimes$ replaces a target side non-terminal D in the first antecedent by the target side γ of the passive rule used to process it.

4.4.3 Example

$p_1 : (N, NN) \xrightarrow{1.0} \langle$ pratique, process $\rangle$	$p_2 : (A/ADJ) \xrightarrow{0.6} \langle$ pratique, helpful $\rangle$
$p_3 : (A/ADJ) \xrightarrow{0.4} \langle$ pratique, practical $\rangle$	$p_4 : (NP/NP) \xrightarrow{1.0} \langle$ l' ingénérie, engineering $\rangle$
$p_5 : (A/NN) \xrightarrow{1.0} \langle$ informatique, software $\rangle$	$p_6 : (NP/NP) \xrightarrow{1.0} \langle$ l' ingénérie A_1, NN_1 engineering $\rangle$
$p_7 : (VP/VP) \xrightarrow{1.0} \langle$ est A_1, is ADJ_1 $\rangle$	$p_8 : (S/SENT) \xrightarrow{1.0} \langle$ NP_1 VP_2, NP_1 VP_2 $\rangle$

Figure 4.34: Example annotated SCFG grammar

$a_1 : \overline{\bullet_{N \rightarrow \bullet} \text{ pratique}, _{NN \rightarrow} \text{process}, 1.0}$	$a_2 : \overline{\bullet_{A \rightarrow \bullet} \text{ pratique}, _{ADJ \rightarrow} \text{helpful}, 0.6}$
$a_3 : \overline{\bullet_{A \rightarrow \bullet} \text{ pratique}, _{ADJ \rightarrow} \text{practical}, 0.4}$	$a_4 : \overline{\bullet_{NP \rightarrow \bullet} \text{l' ingénérie}, _{NP \rightarrow} \text{engineering}, 1.0}$
$a_5 : \overline{\bullet_{A \rightarrow \bullet} \text{informatique}, _{NN \rightarrow} \text{software}, 1.0}$	$a_6 : \overline{\bullet_{NP \rightarrow \bullet} \text{l' ingénérie } A_1, _{NP \rightarrow} NN_1 \text{ engineering}, 1.0}$
$a_7 : \overline{\bullet_{VP \rightarrow \bullet} \text{est } A_1, _{VP \rightarrow} \text{is } JJ_1, 1.0}$	$a_8 : \overline{\bullet_{S \rightarrow \bullet} NP_1\ VP_2, _{SENT \rightarrow} NP_1\ VP_2, 1.0}$

Figure 4.35: Axioms created with annotated SCFG rules.

<table>
<tr><td colspan="5">$[S\bullet \rightarrow NP_1VP_2\bullet, 0, 5]$
$[SENT \rightarrow$ software engineering is practical$, 0.4, 13, (11, 9)]$

$[S\bullet \rightarrow NP_1VP_2\bullet, 0, 5]$
$[SENT \rightarrow$ software engineering is helpful$, 0.6, 12, (11, 8)]$</td></tr>
<tr><td colspan="3">$[\bullet S \rightarrow NP_1 \bullet VP_2, 0, 3]$
$[SENT \rightarrow NP_1VP_2, 1.0, 11, (10)]$

$[NP\bullet \rightarrow$ l’ ingénérie $A_1\bullet, 0, 3]$
$[NP \rightarrow$ software engineering$, 1.0, 10, (7, 1)]$</td><td colspan="2"></td></tr>
<tr><td colspan="2">$[\bullet NP \rightarrow$ l’ ingénérie $\bullet A_1, 0, 2]$
$[NP \rightarrow NN_1$ engineering$, 1.0, 7, (5)]$</td><td></td><td colspan="2">$[VP\bullet \rightarrow$ est $A_1\bullet, 3, 5]$
$[VP \rightarrow$ is practical$, 0.4, 9, (6, 3)]$

$[VP\bullet \rightarrow$ est $A_1\bullet, 3, 5]$
$[VP \rightarrow$ is helpful$, 0.6, 8, (6, 2)]$</td></tr>
<tr><td>$[\bullet NP \rightarrow$ l’ $\bullet$ ingénérie $A_1, 0, 1]$
$[NP \rightarrow NNP_1$ engineering$, 1.0, 5]$

$[\bullet NP \rightarrow$ l’ $\bullet$ ingénérie$, 0, 1]$
$[NP \rightarrow$ engineering$, 1.0, 4]$</td><td></td><td>$[A\bullet \rightarrow$ informatique$\bullet, 2, 3]$
$[NN \rightarrow$ software$, 1.0, 1]$</td><td>$[\bullet VP \rightarrow$ est $\bullet A_1, 3, 4]$
$[VP \rightarrow$ is $ADJ_1, 1.0, 6]$</td><td>$[A\bullet \rightarrow$ pratique$\bullet, 4, 5]$
$[0.4, ADJ \rightarrow$ practical$, 3]$

$[A\bullet \rightarrow$ pratique$\bullet, 4, 5]$
$[0.6, ADJ \rightarrow$ helpful$, 2]$</td></tr>
<tr><td colspan="2">l’</td><td>ingénérie</td><td>informatique</td><td>est</td><td>pratique</td></tr>
<tr><td>0 1</td><td></td><td>1 2</td><td>2 3</td><td>3 4</td><td>4 5</td></tr>
</table>

Figure 4.36: CYK+ chart parsing with annotated SCFG rules.

The axiom a_1 has been removed because its lhs N could not match the input parse tree constituent A.

4.4.4 SCFG Decoding with Target Annotations

Decoding with SCFG grammars annotated on the target language side can be done by removing the source non-terminal labels from the axioms and inference rules given in Equations 4.16, 4.19 and 4.20. The inference rules obtained in this way are given in Equations 4.21, 4.22 and 4.23 where the labeled source non-terminals are simply replaced by X. Note that in this setup, there is no need for the source left-hand-side (lhs) to match the input parse tree. Consequently, only the source rhs of the rules is dotted.

$$\frac{}{[X \rightarrow \bullet\alpha, B \rightarrow \beta, w]} \quad (X, B \xrightarrow{w} \alpha, \beta \in P) \tag{4.21}$$

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, B \rightarrow \beta, i, j, w]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, B \rightarrow \beta, i, j+1, w]} \tag{4.22}$$

$$\frac{[X \rightarrow \alpha_1 \bullet X_n\alpha_2, B \rightarrow \beta_1 D_n \beta_2, i, j, w_1] \quad [X \rightarrow \gamma\bullet, D \rightarrow \delta, j, k, w_2]}{[X \rightarrow \alpha_1 X_n \bullet \alpha_2, B \rightarrow \beta_1\delta\beta_2, i, k, w_1 * w_2]} \tag{4.23}$$

The goal of the system is the item $[X \rightarrow \alpha\bullet, S \rightarrow \beta, 0, |s|, w]$, where the rhs of a rule starting with the start non-terminal (X, S) and spanning the entire input string s has been processed.

4.5 Conclusion

We have presented background knowledge in SCFG-based SMT necessary to understand our contributions to the field, given in Chapters 5, 6 and 7. Our presentation

briefly introduced how SCFG grammars are obtained from parallel data before providing a detailed description of the decoding procedure for SCFG rules.

The contribution given in Chapter 5 is an extension of the SCFG rule extraction and decoding procedures explained throughout this chapter.[13] The contribution in Chapter 6 extends the hierarchical systems explained in Section 4.1 while the contribution in Chapter 7 is an extension of systems with target syntactic annotations, introduced in Sections 4.3.3 and 4.4.4.

[13] A detailed description of the contributions and the sections providing background knowledge to understand them are given at the beginning of Chapter 5.

Chapter 5

Shallow Local Multi Bottom-Up Tree Transducers

We present our contribution to the field of SMT with synchronous grammars, that is the implementation of a system based on the Shallow Local Multi Bottom-Up Tree Transducer (Sh-l-MBOT). A formal definition of l-MBOT has been given in Section 2.1.3. A detailed overview of our contribution can be found in Section 2.3.

We begin our presentation by briefly explaining how to obtain Sh-l-MBOT rules from parallel corpora (Section 5.1). The rule extraction procedures presented here are extensions of SCFG rule extraction, which is explained in Sections 4.1.1 and 4.3. Then we present our main contributions: In Section 5.2, we describe the Sh-l-MBOT translation model and its features. In Section 5.3, we introduce a decoding strategy for Sh-l-MBOT rules without syntactic annotations before showing (in Section 5.4) how to extend it to deal with rules that include syntactic annotations. These contributions are extensions of the decoding procedures for SCFG rules presented in Sections 4.2 and 4.4. In Section 5.5, we present an extensive evaluation of Sh-l-MBOT systems with linguistic annotations at different levels. We conclude this chapter by explaining shortcomings of our approach and possible future work.

5.1 Shallow l-MBOT Grammars

A shallow local multi-bottom-up tree transducer (Sh-l-MBOT) is a restricted variant of the local multi bottom-up tree transducer (l-MBOT) described in Section 2.1.3. The restriction of Sh-l-MBOT requires that the left-hand side (lhs) of each grammar rule has height at most 2. Two procedures for the extraction of Sh-l-MBOT rules have been proposed in the literature:

1. [Seemann et al., 2015a] extend the hierarchical rule extraction heuristic [Chiang, 2007] to extract non-minimal shallow l-MBOT rules. These can either be decorated with syntactic annotation or not. When using syntactic annotation, the rules can be decorated on both source and target language side or on one side only.

2. [Maletti, 2011] presents an algorithm to extract minimal l-MBOT rules with syntactic annotation on the source and target language side. In [Braune et al., 2013] the obtained rules are made shallow by removing the internal nodes of the trees composing these rules.

5.1.1 Sh-l-MBOT rules without syntactic annotation

The extraction procedure for non-minimal Sh-l-MBOT rules without syntactic annotation in [Seemann et al., 2015a] extends the heuristic in [Chiang, 2007] to extract hierarchical rules with multiple target language sides. The input to rule extraction are word aligned sentence pairs as shown in Figure 5.1.

The rule extraction heuristic extends the notion of “initial phrase pairs” [Chiang, 2007, p.266] to ”initial rules” that are defined as spans $\langle p, p_1 \dots p_n \rangle$ on the input sentences. The target spans $p_1 \dots p_n$ are the discontiguous target sides of the undecorated Sh-l-MBOT rules. Initial rules can be extracted from a parallel sentence pair if the following conditions hold:

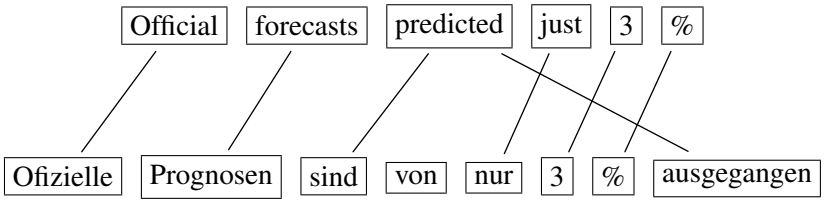


Figure 5.1: Word aligned sentence pair

Condition 1	There is at least one aligned word between the source words in span p and the target words in spans $p_1 \dots p_n$.
Condition 2	All words aligned to the source words in span p are in the target segments in spans $p_1 \dots p_n$.
Condition 3	All words aligned to the target segments in spans $p_1 \dots p_n$ are in the source language phrase p.

Figure 5.2: Heuristic to extract initial Sh-l-MBOT rules

Once initial rules have been extracted, further rules can be obtained by subtracting smaller initial rules from bigger ones. To reduce the number of extracted rules, [Chiang, 2007] restricts the rule extraction procedure as presented in Section 4.1.1. [Seemann et al., 2015a] apply the same restrictions to their heuristic. Example Sh-l-MBOT rules extracted from the parallel sentences in Figure 5.1 are r_1 to r_9 below. In our representation of the rules below we write the words covered by the spans $\langle p, p_1 \dots p_n \rangle$ instead of the spans. We use the symbol ";" to separate target sequences.

r_1 $X \rightarrow \langle$ predicted just 3 %, sind von nur 3 % ausgegangen $\rangle$

r_2 $X \rightarrow \langle$ predicted, sind ; ausgegangen $\rangle$

r_3 $X \rightarrow \langle$ predicted just, sind ; nur ; ausgegangen $\rangle$

r_4 $X \rightarrow \langle$ predicted just, sind von nur ; ausgegangen $\rangle$

r_5 $X \rightarrow \langle$ X_1 just 3 %, X_1 ; von nur 3 % X_1 $\rangle$

r_6 $X \rightarrow \langle$ X_1 just X_2, X_1 ; von nur X_2 X_1 $\rangle$

r_7 $X \rightarrow \langle$ predicted just X_1, sind ; nur ; X_1 ; ausgegangen $\rangle$

r_8 $X \rightarrow \langle$ predicted just X_1, sind von nur ; X_1 ; ausgegangen $\rangle$

r_9 $X \rightarrow \langle$ X_1 forecasts, X_1 Prognosen $\rangle$

5.1.2 Sh-l-MBOT rules with syntactic annotations on both sides

Sh-l-MBOT rules with syntactic annotations on both sides are extracted from a bi-parsed and word aligned parallel corpus as shown in Figure 4.17 (Section 4.3). Rules with source and target side syntactic annotation can either be extracted (i) following [Seemann et al., 2015a] which yields a set of (potentially) non-minimal rules or (ii) following [Maletti, 2011] which yields a set of minimal Sh-l-MBOT rules.

Non-minimal Rules

Non-minimal Sh-l-MBOT rules are generated by adding the following constraints to the heuristic presented in Section 5.1.1:

4. each target span p_1 to p_n matches a syntactic constituent in the target language parse tree
5. each source span p matches a syntactic constituent in the source language parse tree

Example Sh-l-MBOT rules extracted following this procedure are given in Figures 5.3 and 5.4.

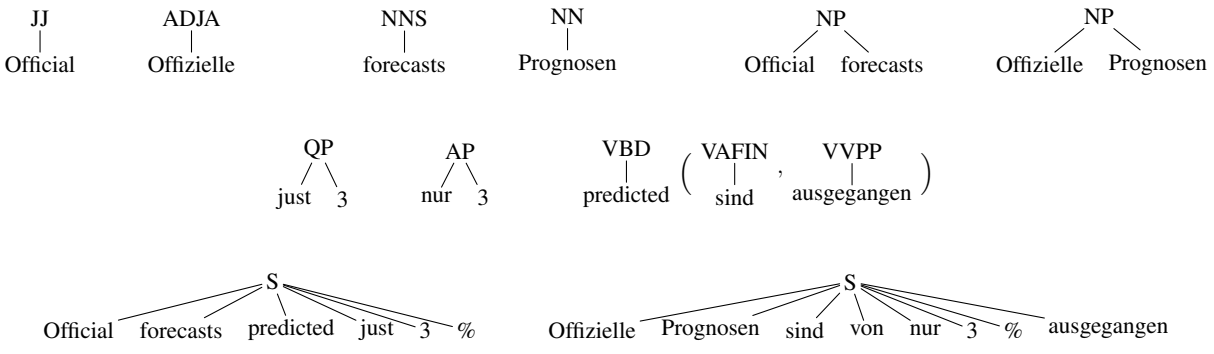


Figure 5.3: Initial Rules extracted by the Sh-l-MBOT heuristic.

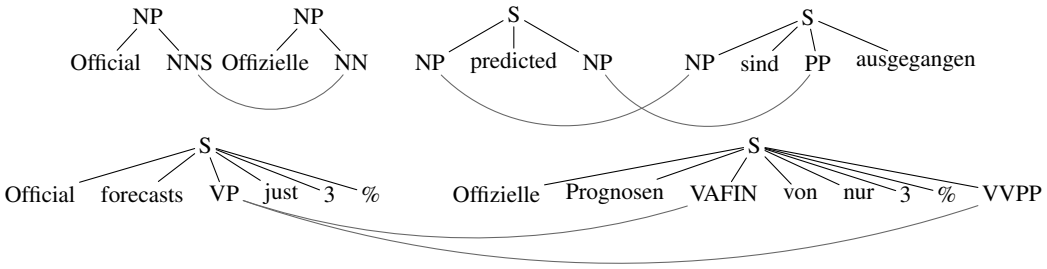


Figure 5.4: Sh-l-MBOT rules obtained after subtraction step.

Minimal Rules

Minimal Sh-l-MBOT rules can be extracted according to the procedure summarized in Figure 5.5. Unlike STSG rules, Sh-l-MBOT rules allow sequences of trees on the target language side. This difference implies that in Step 2 of the rule extraction procedure, several target nodes are allowed to correspond to a given maximal source node, in contrast to the STSG rule extraction (Section 4.3.4), where only single target nodes were considered. We illustrate the Sh-l-MBOT rule extraction in Figures 5.6 and 5.9. Figures 5.10 and 5.11 show the rules extracted after the 2 first passes of the algorithm. The illustrations show that unlike the STSG rule extraction procedure, all words, including the pair *predicted-sind ausgegangen*, could be extracted in the first pass of the Sh-l-MBOT rule extraction.

Step 1	Find a maximal source node consistent with a (minimal) set S of word alignment links
Step 2	Find maximal **target nodes** consistent with the minimal extension of S *Reiterate 1 and 2 if necessary*
Step 3	Excise the rule with maximal source and **target nodes** from the biparsed sentence pair and replace with parent labels
Step 4	Add the rule to the rule set
Step 5	Repeat

Figure 5.5: Main steps of the rule extraction procedure for minimal Sh-l-MBOT rules.

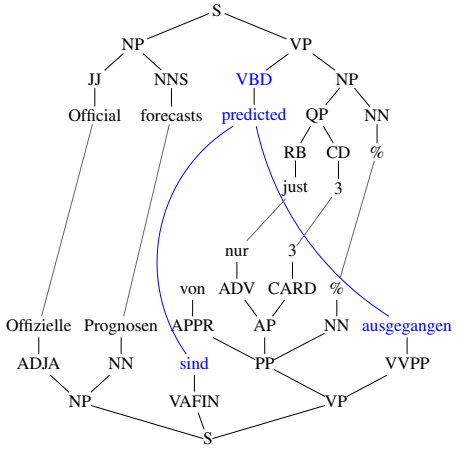
Step 1	**Find maximal source node consistent with word alignment links**
Algorithm	Given a lowest node l_s in the source tree that is not a non-terminal leaf and that is aligned to a set of leaves $\{l_{t1}, ..., l_{tn}\}$ in the target tree, find the highest node in the source tree such that no leaf in the subtree aligns to $l_{tk} \notin \{l_{t1}, ..., l_{tn}\}$.
Example	We assume that the English tree is the source tree. The source leaf node $l_s = predicted$ is aligned to the target leaf nodes $l_{t1} = sind$ and $l_{t2} = ausgegangen$. The maximal node in the source tree that still aligns to l_{t1} and l_{t2} is VBD. 

Figure 5.6: First step of Sh-l-MBOT rule extraction procedure.

5.1.3 Sh-l-MBOT rules with target syntactic annotations

Shallow l-MBOT rules with syntactic annotation on the target language side are extracted from pairs of word aligned strings and trees. An example is given in Figure 4.31 (Section 4.3.5). We give another example in Figure 5.12.

Sh-l-MBOT rules with syntactic annotation on the target language side are obtained in the same way as rules with annotations on both sides (Section 5.1.2) except that constraint (4) above is dropped. Example Sh-l-MBOT rules extracted following this procedure are given in Figures 5.13 and 5.14. We represent the non-annotated source side of the rules with trees rooted at the non-terminal X.

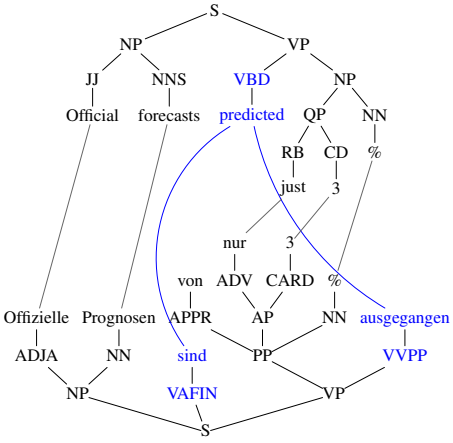
Step 2	**Find maximal target nodes consistent with word alignment links**
Algorithm	Given a set of target leaves $\{l_{t1}, ..., l_{tn}\}$ in the target tree that is aligned to the source leaves $\{l_{s1}, ..., l_{sm}\}$, find the highest nodes in the target tree such that they contain all leaves $\{l_{t1}, ..., l_{tn}\}$.
Example	We assume that the English tree is the source tree. The target leaf nodes are $l_{t1} = sind$ and $l_{t2} = ausgegangen$. The source leaf node is $l_s = predicted$. The maximal nodes in the target tree that still align to l_s are $VAFIN$ and $VVPP$. 

Figure 5.7: Second step of Sh-l-MBOT rule extraction procedure.

5.2 The Shallow l-MBOT Translation Model

As for hierarchical machine translation, many Sh-l-MBOT rules can be applied to an input sentence to derive different output sentences. In order to predict the best translation e of an input sentence f, [Braune et al., 2013] define a translation model for Sh-l-MBOT rules.

5.2.1 Mathematical Definition

The Shallow l-MBOT translation model is a log-linear model [Och and Ney, 2002] over the derivations D. The probability $P(D)$ of a derivation $D = (S_s, S_t, \widetilde{A}) \underset{r_1}{\Rightarrow} (T_{p1}, T_{s2}, \widetilde{A}) \underset{r_2}{\Rightarrow} ... \underset{r_l}{\Rightarrow} \langle f, e \rangle$ is defined as the weighted product of the features in this derivation.

Step 3	**Excise the rule with maximal source and target nodes**
	Remove the maximal node in the source tree as well as its subtree. Replace the removed subtree by its root. Remove the maximal nodes in the target tree as well as their subtrees. Replace each removed subtree by its root. Align the remaining non-terminals.
Example	Remove the source subtree rooted at VBD and replace with VBD. Remove the subtrees rooted at $VAFIN$ *and* $VVPP$ and replace with $VAFIN$ *and* $VVPP$. Align VBD with $VAFIN$ and $VVPP$.

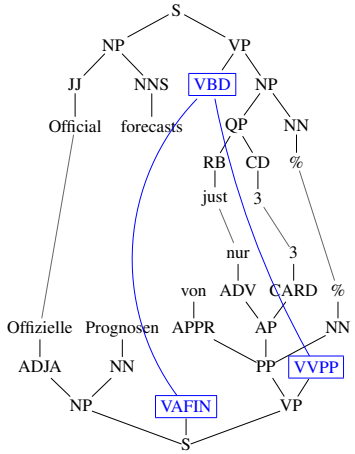


Figure 5.8: Third step of Sh-l-MBOT rule extraction procedure.

Step 4	**Add rule to rule set**
Example	VBD (predicted) (VAFIN (sind) , VVPP (ausgegangen))

Figure 5.9: Fourth step of Sh-l-MBOT rule extraction procedure.

$$P(D) \propto LM(e)^{\lambda_m} \prod_{i=1}^{m-1} h_i(D)^{\lambda_i} \tag{5.1}$$

where D denotes an l-MBOT derivation and $LM(e)$ is the evaluation of the language model on e. We assume that the language model is the m-th feature. As seen in Section 2.1.3, the weight of an l-MBOT derivation is the product of the rules used

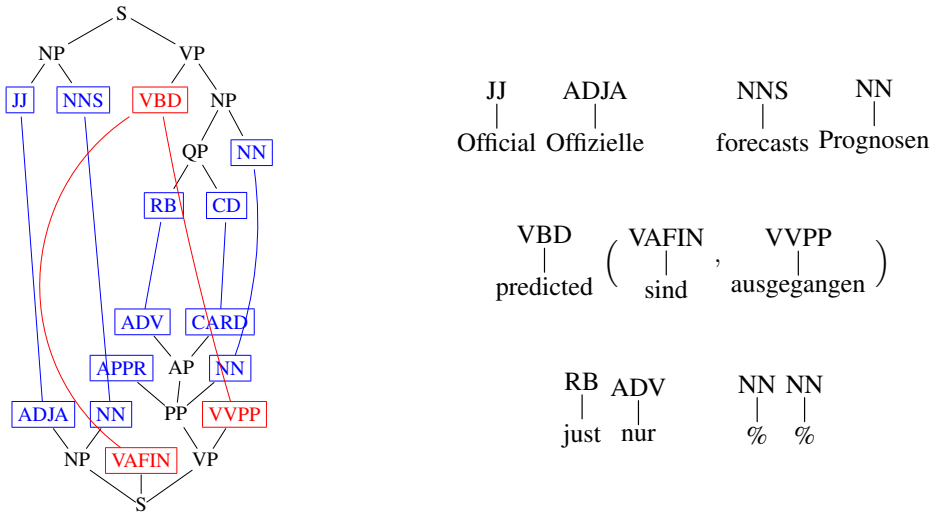


Figure 5.10: Word aligned biparsed sentence pair and extracted Sh-l-MBOT rules after the first pass of the algorithm.

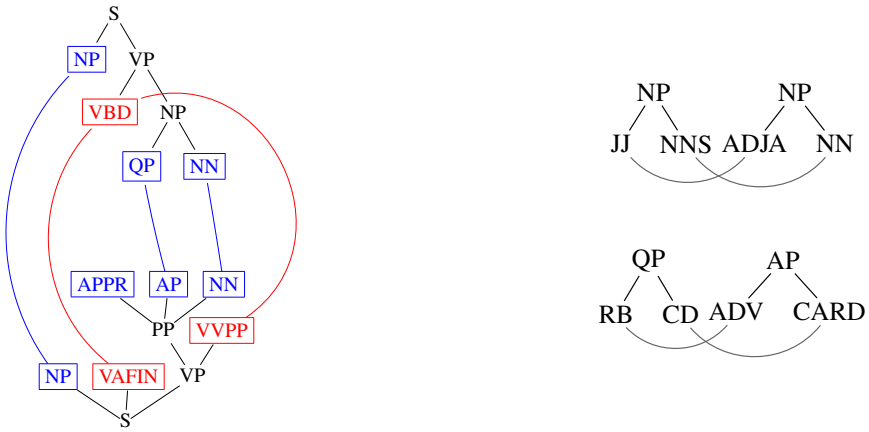


Figure 5.11: Word aligned biparsed sentence pair and extracted Sh-l-MBOT rules after the second pass of the algorithm.

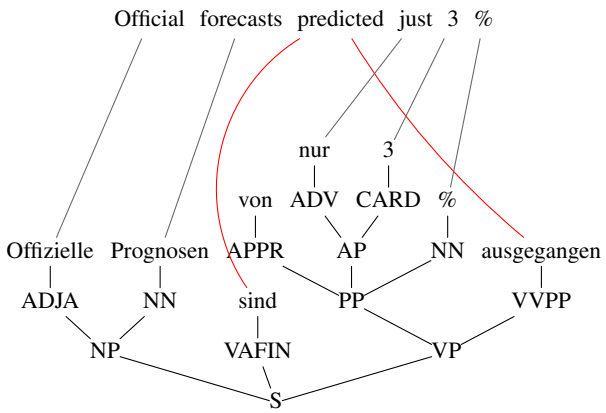


Figure 5.12: Word aligned sentence pair with target parse tree.

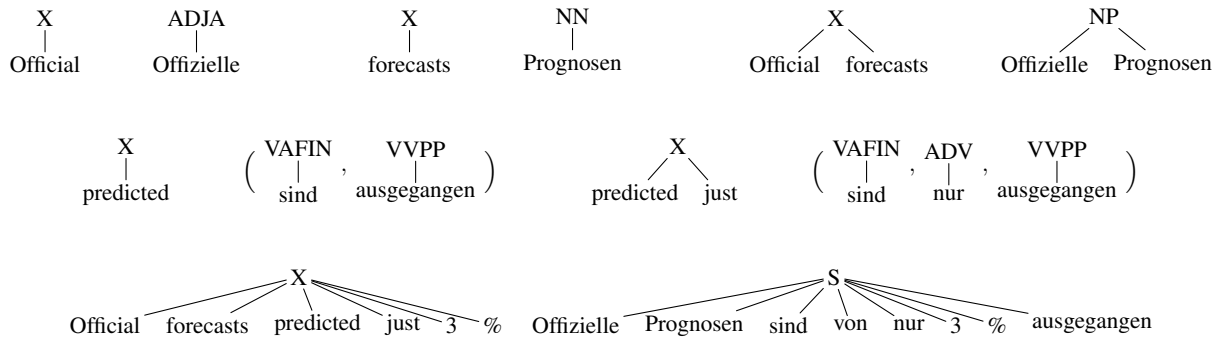


Figure 5.13: Initial rules extracted by the Sh-l-MBOT heuristic.

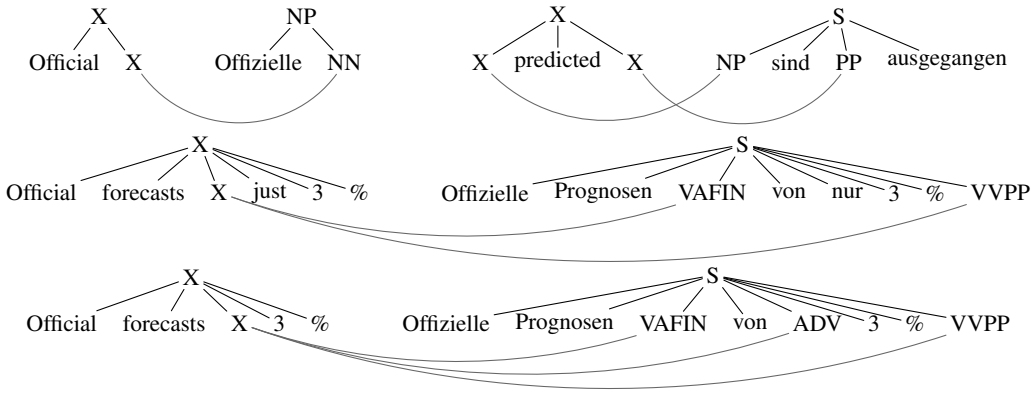


Figure 5.14: Sh-l-MBOT rules obtained after subtraction step.

in this derivation. Hence, Equation 5.1 can be written as:

$$P(D) \propto LM(e)^{\lambda_m} \prod_{i=1}^{m-1} \prod_{j=1}^{l} h_i(r_j)^{\lambda_i} \tag{5.2}$$

5.2.2 Model Features

The estimate of the language model feature $LM(e)$ has been adapted to score partial translations consisting of discontiguous units. The details are explained in Section 5.3.3. The features on Sh-l-MBOT rules include:

1. The indirect translation weight of a rule, which defines the probability $P(T_1 \mid T_{s2}, \widetilde{A})$ to see the source tree T_1 given the target tree sequence T_{s2}.
2. The direct translation weight of a rule, which defines the probability $P(T_{s2} \mid T_1, \widetilde{A})$ to see the target tree sequence T_{s2} given the source tree T_1.
3. The indirect lexical weight of a rule, which defines the probability $Pw(T_1 \mid T_{s2}, \widetilde{A})$ to see the terminal leaves of T_1 given the terminal leaves of T_{s2}.
4. The direct lexical weight of a rule, which defines the probability $Pw(T_{s2} \mid T_1, \widetilde{A})$ to see the leaves labeled with terminal symbols (terminal leaves) of T_{s2} given the terminal leaves of the source tree T_1.
5. A constant used to count the number of rules used in a derivation.
6. A constant used to count the number of glue rules used in a derivation.
7. A constant used to count the number of (potentially) non-contiguous glue rules used in a derivation.
8. The number of terminal symbols in a rule.
9. The number of components in the target side sequence.

5.2.3 Feature Training

Translation weights

The indirect translation weight $P(T_1 \mid T_{s2}, \widetilde{A})$ is computed through relative frequency estimation. More precisely, the frequency of the rule is divided by the sum of the frequencies of all rules with the same rhs. We illustrate (in Figure 5.15) this scoring procedure with rules having syntactic annotations on the source and target language side but it also applies to non-annotated or target-annotated Sh-l-MBOT rules. The frequency is indicated at the right of each rule. Assuming that the displayed rules are the only ones with the shown lhs and rhs, each rule gets the indirect translation score $P(T_1 \mid T_{s2}, \widetilde{A}) = 1$. The direct translation weight $P(T_{s2} \mid T_1, \widetilde{A})$ is obtained by dividing the frequency of the rule by the sum of the frequencies of the rules with the same lhs. This yields the following scores: rule r_1 gets $\frac{15}{30} = \frac{1}{2}$, rule r_2 gets $\frac{10}{30} = \frac{1}{3}$ and rule r_3 gets $\frac{5}{30} = \frac{1}{6}$.

r_1 VBD(predicted) (VAFIN(sind) , VVPP(ausgegangen)) 15

r_2 VBD(predicted) (VAFIN(haben) , VVPP(vorausgesagt)) 10

r_3 VBD(predicted) VAFIN(prognostizierten) 5

Figure 5.15: Sh-l-MBOT rule with the same lhs

In addition to relative frequency estimation, an elementary discounting of rare rules has been performed by multiplying the rule weight of rules extracted at most 10 times by 10^{-2}.

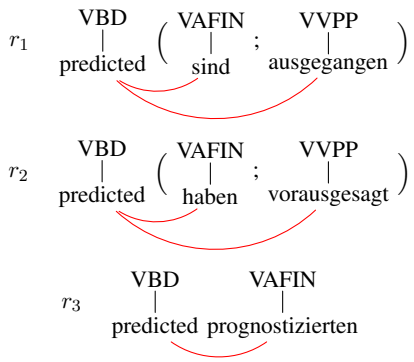
Lexical weights

The lexical weights are obtained by multiplying the lexical probabilities of aligned words in a rule, across all trees in the target side sequence. Each time a word is aligned to multiple words, the average of the lexical probabilities is taken. The computation of the indirect lexical weight is given by $Pw(v \mid u) = \prod_{j=1}^{|v|} \frac{1}{|\{i|(i,j)\in A\}|} \sum_{(i,j)\in A} lex(v_j \mid u_i)$, where (i) v and u denote the string of the terminal symbols in the leaves of T_1 and T_{s2}, (ii) A stands for the word alignment between the leaves of T_1 and T_{s2}, (iii) $lex(v_j \mid u_i)$ denotes the word translation probability of the j-th word in v given the i-th word in u. In the same fashion, the direct lexical weight is computed as $Pw(u \mid v) = \prod_{i=1}^{|u|} \frac{1}{|\{j|(i,j)\in A_j|} \sum_{(i,j)\in A} lex(u_i \mid v_j)$. Unaligned words are considered as being aligned to a special empty word, which also has lexical translation probabilities. To illustrate the training of lexical weights, consider rules r_1 to r_3. The alignments between terminal symbols in these rules are given in Figure 5.16. The indirect lexical weight of r_1 is $Pw(v \mid u) = \frac{0.01+0.15}{2} = 0.08$. The direct lexical weight is $Pw(u \mid v) = 0.01 * 0.15 = 0.0015$. For rule r_2, we have $Pw(v \mid u) = \frac{0.05+0.30}{2} = \frac{0.35}{2}$ and $Pw(u \mid v) = 0.05 * 0.30 = 0.015$. Finally, for r_3 the direct and indirect lexical weights are 0.2.

The example above shows that the estimation of lexical probabilities for shallow l-MBOT rules is suboptimal. The translations *sind ausgegangen* and *haben vorausgesagt* of *predicted* are at least as good as *prognostizierten* but their lexical probabilities are much lower. Note that because they model discontiguous lexical units all syntax-based approaches potentially have the same problem. For instance, the lexical scores of the hierarchical rule $X \rightarrow \langle$ predicted X, haben X vorausgesagt $\rangle$ is the same as for rule r_3.

Count features

The count features 5 to 8 in the translation model assign a negative weight or penalty to the elements they are defined on, in the same way as for the hierarchi-



English Word	German Word	Weight
predicted	sind	0.01
predicted	ausgegangen	0.15
predicted	prognostizierten	0.20
predicted	haben	0.05
predicted	voausgesagt	0.30

Figure 5.16: Lexical weight of aligned words in rules r_1 to r_3.

cal model. The count c of target fragments in the sequence is scored as 100^{1-c} to discourage rules with many fragments.

Weight training

The weights of the features in the hierarchical translation model are trained using minimum error rate training [Och, 2003].

5.3 Decoding without syntactic annotation

A rule extraction procedure yielding Sh-l-MBOT rules without syntactic annotation has been presented in Section 5.1.1. In order to build a SMT system using these rules, the hierarchical decoding procedure presented in Section 4.2 has to be extended to deal with rules having (possibly) discontiguous target sides.

5.3.1 CYK+ parsing and Translation Generation

As opposed to the hierarchical case, decoding with Sh-l-MBOT rules requires to integrate the target side of the rules into the deductive inference process presented in Section 4.2.1. This needs to be done because Sh-l-MBOT rules with multiple target side components $\gamma_1, \ldots, \gamma_n$ can only be plugged into rules with the same

number of target non-terminals $X_1, \ldots, X_n$ aligned to the same source non-terminal X. To illustrate this point, consider the Sh-l-MBOT grammar in Figure 5.17. Rule r_1, which has 3 target side components can only be composed with rule r_7, which has the same number of target non-terminals (X_1). In the same way, rule r_2 can only be assembled with rule r_5.

r_1 : X $\xrightarrow{0.6}$ ⟨ predicted, sind ; von; ausgegangen ⟩	r_2 : X $\xrightarrow{0.3}$ ⟨ predicted, haben ; vorausgesagt ⟩
r_3 : X $\xrightarrow{0.1}$ ⟨ predicted, prognostizierten ⟩	r_4 : X $\xrightarrow{1.0}$ ⟨ increase, Erhöhung ⟩
r_5 : X $\xrightarrow{0.5}$ ⟨ They X_1 an X_2, Sie X_1 eine X_2 X_1 ⟩	r_6 : X $\xrightarrow{0.2}$ ⟨ They X_1 an X_2, Sie X_1 eine X_2 ⟩
r_7 : X $\xrightarrow{0.3}$ ⟨ They X_1 an X_2, Sie X_1 X_1 einer X_2 X_1 ⟩	

Figure 5.17: Example Sh-l-MBOT grammar without syntactic annotations

In order to capture this specificity, the deductive proof system in Section 4.2.1 can be extended in two distinct ways:

1. By explicitly representing the target side of the rules in the axioms and parse items.
2. By using the same items as for hierarchical decoding and only taking the target sides into account to restrict non-lexical inference.

Although option (2) is simpler and closer to the proof system already presented in Section 4.2.1, we find that (1) leads to a more generic description because it already integrates all components required to define translation generation.

CYK+ parsing for Sh-l-MBOT grammars

The axioms of the Sh-l-MBOT parser can be written as:

$$\frac{}{[X \rightarrow \bullet\alpha, w, \gamma_1 \ldots \gamma_n]} \quad (X \xrightarrow{w} \langle \alpha, \gamma_1 \ldots \gamma_n \rangle \in P) \tag{5.3}$$

where $X \xrightarrow{w} \langle \alpha, \gamma_1 \ldots \gamma_n \rangle$ is a Sh-l-MBOT rule with target side sequence $\gamma_1 \ldots \gamma_n$. Lexical inference rules can be written as:

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w, \gamma_1 \ldots \gamma_n]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w, \gamma_1 \ldots \gamma_n]} \tag{5.4}$$

In these rules, the consequent simply carries the same sequence of partial translation as the antecedent. Non-lexical inference rules, given in Equation 5.5, are more constrained. In these rules, a non-terminal X in an active parse item is processed using a passive item provided that the number of target non-terminals X in the active item is the same as the number n of target side components in the passive item.

$$\frac{[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_{11} \ldots \gamma_{1n}] \quad [X \rightarrow \beta\bullet, j, k, w_2, \gamma_{21} \ldots \gamma_{2m}]}{[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2, \gamma_{11} \ldots \gamma_{1n} \otimes \gamma_{21} \ldots \gamma_{2m}]}, * \tag{5.5}$$

where $*$ means that the number of non-terminals in $\gamma_{11} \ldots \gamma_{1n}$ is limited to m. On the target language side of these items, all non-terminals in the sequence $\gamma_{11} \ldots \gamma_{1n}$ that are aligned to X are replaced by the sequence $\gamma_{21} \ldots \gamma_{2m}$ in left to right order. This operation is denoted by $\otimes$. Translation generation is included in the inference rules above. Following the terminology adopted in Section 4.2.2, we call the target language sides of passive parse items *translation options*. In Sh-l-MBOT parsing, translation options can consist of several discontiguous segments. The goal of the system is the item $[S \rightarrow \alpha\bullet, 0, |s|, w, \beta]$, where the rhs of a rule starting with the start non-terminal S and spanning the entire input string s has been processed. When the goal is reached, i.e, when the entire input sentence has been processed, there are no target side discontiguities anymore.

Because the source side of Sh-l-MBOT rules is the same as for hierarchical rules, the CYK+ search procedure and the n-best list generation for Sh-l-MBOT grammars are the same as those given in Algorithms 1 and 2.

5.3.2 Example

We illustrate CYK+ parsing with the Sh-l-MBOT grammar in Figure 5.17 as well as the input sentence E :

E *They predicted an increase*

From these Sh-l-MBOT rules, the axioms in Figure 5.18 are created following Equation 5.3. As seen in Section 5.3.1, the axioms can:

(i) carry multiple target language sides $\gamma_1, \ldots, \gamma_n$ (such as the sequence *sind; von; ausgegangen* in axiom a_1).

(ii) contain multiple target non-terminals aligned to a single source side non-terminal (such as three X_1 in axiom a_7)

a_1 : $\overline{X \rightarrow \bullet \text{ predicted}, 0.6, \text{ sind ; von ; ausgegangen}}$	a_2 : $\overline{X \rightarrow \bullet \text{ predicted}, 0.3, \text{ haben ; vorausgesagt}}$
a_3 : $\overline{X \rightarrow \bullet \text{ predicted}, 0.1, \text{ prognostizierten}}$	a_4 : $\overline{X \rightarrow \bullet \text{ increase}, 1.0, \text{Erhöhung}}$
a_5 : $\overline{X \rightarrow \bullet \text{They } X_1 \text{ an } X_2, 0.5, \text{ Sie } X_1 \text{ eine } X_2 \ X_1}$	a_6 : $\overline{X \rightarrow \bullet \text{They } X_1 \text{ an } X_2, 0.2, \text{ Sie } X_1 \text{ eine } X_2}$
a_7 : $\overline{X \rightarrow \bullet \text{They } X_1 \text{ an } X_2, 0.3, \text{ Sie } X_1 \ X_1 \text{ einer } X_2 \ X_1}$	

Figure 5.18: Axioms created with rules having lhs X into $a[X]$

During CYK+ chart parsing, rules are assembled according to the combination operator $\otimes$ defined in Section 5.3.1. Figure 5.19 shows the items inferred from the axioms in Figure 5.18 on increasing spans of sentence E. For better readability we only display the target sides $\gamma_1 \ldots \gamma_n$ of passive rules (the translation options). This illustration shows, for instance, that the passive item $[X \rightarrow \text{ predicted} \bullet, 1, 2, \ 0.6, \text{ sind ; von ; ausgegangen}]$ cannot be combined with $[X \rightarrow \text{ They} \bullet X_1 \text{ an } X_2, 0, 1, \ 0.2, \text{Sie } X_1 \text{ eine } X_2]$ because the number of its target sides (3) is different from the number (1) of aligned target non-terminals in the active item. The translation options *sind ; von ; ausgegangen* and *haben ; vorausgesagt* covering span 1 are discontiguous.

5.3.3 Language Model Integration

When parsing with Sh-l-MBOT grammars, translation options can be discontiguous. This means that instead of a contiguous target language segment γ, a sequence of

discontiguous segments $\gamma_1 \dots \gamma_n$ are passed to the language model scoring functions. To LM-score such segments, we define the function PS_{LM} which applies to discontiguous segments of terminals $\gamma_1 \dots \gamma_n$ and multiplies the language model score for each segment. PS_{LM} can be written as:

$$PS_{LM}(\gamma_1 \dots \gamma_n) = \prod_{i=1}^{n} P_{LM}(\gamma_i) \tag{5.6}$$

where P_{LM} is defined as in Equation 4.12. For each segment γ_i bigger than the size m of the m-gram language model, we apply the function $Mark_m$ defined in Equation 4.13. The function $PS_{LM}(\gamma_1 \dots \gamma_n)$ yields unprecise language model scores for translation options consisting of many small discontiguous units because it treats these as being independent although those will be combined into a single string. However, as soon as segments in $\gamma_1 \dots \gamma_n$ are assembled to form larger units, these are scored. This means that the LM scores get more accurate as the span size increases. As in the final rules only one component is allowed, the final LM score is computed for the complete output sentence.

The functions $PS_{LM}(\gamma_1 \dots \gamma_n)$ and $Mark_m(\gamma)$ are added to the Sh-l-MBOT inference rules. The axioms become:

$$\frac{}{[X \to \bullet\alpha, w, \gamma_1 \dots \gamma_n, Mark_m(\gamma_i), PS_{LM}(\gamma_1 \dots \gamma_n)]} \quad (X \xrightarrow{w} \langle \alpha, \gamma_1 \dots \gamma_n \rangle \in P) \tag{5.7}$$

The lexical inference rule becomes:

$$\frac{[X \to \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w, \gamma_1 \dots \gamma_n, Mark_m(\gamma_i), PS_{LM}(\gamma_1 \dots \gamma_n)]}{[X \to \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w, Mark_m(\gamma_i), PS_{LM}(\gamma_1 \dots \gamma_n)]} \tag{5.8}$$

The non-lexical inference rule is given by the following equations where 5.9 and 5.10 are the antecedents of the item and 5.11 is the consequent.

$$[X \to \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_{11} \dots \gamma_{1n}, Mark_m(\gamma_{1i},), PS_{LM}(\gamma_{11} \dots \gamma_{1n})] \tag{5.9}$$

$$[X \rightarrow \beta\bullet, j, k, w_2, \gamma_{21} \ldots \gamma_{2k}, Mark_m(\gamma_{2j},), PS_{LM}(\gamma_{21} \ldots \gamma_{2k})] \tag{5.10}$$

$$[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2, Mark_m(\gamma_{1i} \otimes \gamma_{2j}), PS_{LM}(\gamma_{11} \ldots \gamma_{1n} \otimes \gamma_{21} \ldots \gamma_{2k})] \tag{5.11}$$

Figure 5.20 illustrates language model scoring with a 4-gram language model using $PS_{LM}(\gamma_1 \ldots \gamma_n)$ on the translation options in Figure 5.23. The example shows that for the discontiguous sequences *sind; von; ausgegangen* and *haben; vorausgesagt*, language model scoring is applied to all components of the translation options. The scores obtained there are quite unprecise, as the components of the sequences are treated as independent unigrams. But later on, these units are assembled into contiguous segments such as *Sie sind von einer Erhöhung ausgegangen* and the language model score is recomputed for such larger segments. However, due to the inexact language model estimation, good parse items are subject to be pruned out due to their inexact language model score.

With these new inference rules, the CYK+ search algorithm as well as cube pruning can be performed as described in Sections 4.2.1 and 4.2.6.

5.4 Decoding with syntactic annotation

The decoding procedure for unannotated Sh-l-MBOT grammars can be extended to handle grammars with syntactic annotation on the source and target language side or on the target language side only. We begin by presenting decoding with syntactic annotations on the source and target language side.

5.4.1 Source and Target Syntactic Annotations

Decoding with annotated Sh-l-MBOT rules is done by augmenting the items in Equations 5.3, 5.4 and 5.5 with labels taken from source and target non-terminal alphabets. In this setup, the axioms of the Sh-l-MBOT chart parser have the form:

$$\frac{}{[\bullet A \rightarrow \bullet\alpha, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, w]} \quad (A, B_1, \ldots, B_n) \xrightarrow{w} \langle \alpha, \beta_1 \ldots \beta_n \rangle \in P \tag{5.12}$$

where A is the labeled source left-hand-side (lhs) of the Sh-l-MBOT rule and $B_1 \ldots B_n$ are the head labels of the components in the target lhs of the rule. The dots indicate that both, the source lhs A and the source string α remain to be recognized for the subtree rooted at A to be processed. Passive parse items have the form in Equation 5.13, where the non-terminal A is recognized if it matches the input parse tree.

$$[A\bullet \rightarrow \alpha\bullet, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w] \tag{5.13}$$

The inference rule in Equation 5.14 specifies how non-terminals in the source lhs of dotted rules are processed. $V_{i,j}$ denotes parse labels of the input sentence spanning from i to j.

$$\frac{[\bullet A \rightarrow \alpha_1\bullet, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w]}{[A\bullet \rightarrow \alpha_1\bullet, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w]} \quad A \in V_{i,j} \tag{5.14}$$

Lexical inference rules, given in Equation 5.15 work in the same fashion as Sh-l-MBOT rules without syntactic annotation. They simply consume a terminal symbol t_{j+1} spanning from j to $j+1$.

$$\frac{[\bullet A \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w]}{[\bullet A \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j+1, w]} \tag{5.15}$$

Non-lexical inference rules, given in Equation 5.16, process source and target non-terminals that are labeled. When a passive parse item such as $[C\bullet \rightarrow \gamma\bullet, D_1 \dots D_n \rightarrow \delta_1 \dots \delta_n, j, k, w_2]$ is used to recognize a source non-terminal in an active item, its source lhs C as well as its target side components $D_1 \dots D_n$ must match the source non-terminal of the active item as well as all the target non-terminals aligned to it.

$$\frac{[\bullet A \rightarrow \alpha_1 \bullet C_m \alpha_2, B_1, \dots, B_n \rightarrow \beta_{11} D_{1m} \beta_{12} \dots \beta_{l1} D_{lm} \beta_{l2}, i, j, w_1] \quad [C\bullet \rightarrow \gamma\bullet, D_1 \dots D_l \rightarrow \delta_1 \dots \delta_l, j, k, w_2]}{[\bullet A \rightarrow \alpha_1 C_m \bullet \alpha_2, B_1, \dots, B_n \rightarrow \beta_{11} \delta_1 \beta_{12} \dots \beta_{l1} \delta_l \beta_{l2}, i, k, w_1 * w_2]} \tag{5.16}$$

The goal of the system is the item $[S\bullet \rightarrow \alpha\bullet, S \rightarrow \beta, 0, |s|, w]$, where the rhs of a rule starting with the start non-terminal S and spanning the entire input string has been processed.

Given these items, the parsing procedure is the same as for non-annotated Sh-l-MBOT grammars.

5.4.2 Example

We now illustrate CYK+ parsing with Sh-l-MBOT grammars having source and target syntactic annotations. We decode the parsed English sentence E in Figure 5.21 using the axioms in Figure 5.22. As opposed to the non-annotated case, the axioms created from annotated Sh-l-MBOT rules carry the head labels of the target side components as well as the labels of the source lhs.

During CYK+ chart parsing, the input sentence is processed according to the inference rules in Equations 5.14, 5.15 and 5.16. As opposed to non-annotated decoding, the labels of the source parse tree must be matched for parse items to become passive. For instance, the item $[VBD\bullet \rightarrow \text{predicted}\bullet, 1, 2]$ could only be created because the span 1-2 in the input parse tree carries the label VBD.

<table>
<tr><td colspan="4">[$X \rightarrow$ They X_1 an X_2 •, 0, 4]
[Sie sind von einer Erhöhung ausgegangen, 0.18, 16, (13, 4)]

[$X \rightarrow$ They X_1 an X_2 •, 0, 4]
[Sie haben eine Erhöhung vorausgesagt, 0.15, 15, (12, 4)]

[$X \rightarrow$ They X_1 an X_2 •, 0, 4]
[Sie prognostizierten eine Erhöhung, 0.02, 14, (11, 4)]</td></tr>
<tr><td colspan="3">[$X \rightarrow$ They X_1 an • X_2 , 0, 3]
[0.18, 13, (10)]

[$X \rightarrow$ They X_1 an • X_2 , 0, 3]
[0.15, 12, (9)]

[$X \rightarrow$ They X_1 an • X_2, 0, 3]
[0.02, 11, (8)]</td><td></td></tr>
<tr><td colspan="2">[$X \rightarrow$ They X_1 • an X_2, 0, 2]
[0.18, 10, (7, 3)]

[$X \rightarrow$ They X_1 • an X_2, 0, 2]
[0.15, 9, (6, 2)]

[$X \rightarrow$ They X_1 • an X_2, 0, 2]
[0.02, 8, (5, 1)]</td><td></td><td></td></tr>
<tr><td>[$X \rightarrow$ They • X_1 an X_2, 0, 1]
[0.3, 7]

[$X \rightarrow$ They • X_1 an X_2, 0, 1]
[0.5, 6]

[$X \rightarrow$ They • X_1 an X_2, 0, 1]
[0.2, 5]</td><td>[$X \rightarrow$ predicted•, 1, 2]
[0.6, sind ; von ; ausgegangen, 3]

[$X \rightarrow$ predicted•, 1, 2]
[0.3, haben ; vorausgesagt, 2]

[$X \rightarrow$ predicted•, 1, 2]
[0.1, prognostizierten, 1]</td><td></td><td>[$X \rightarrow$ increase•, 3, 4]
[1.0, Erhöhung, 4]</td></tr>
<tr><td>They</td><td>predicted</td><td>an</td><td>increase</td></tr>
<tr><td>0 1</td><td>1 2</td><td>2 3</td><td>3 4</td></tr>
</table>

Figure 5.19: Active (in red at top) and passive items created during parsing with Sh-l-MBOT.

<table>
<tr><td colspan="4">[$X \rightarrow$ They X$_1$ an X$_2$ $\bullet, 0, 1$]
[Sie sind von einer Erhöhung ausgegangen, 0.18, $(7, 3, 4)$]
[P_{LM}(Sie sind von einer Erhöhung ausgegangen)]
[$Mark_m$(Sie sind von einer Erhöhung ausgegangen)]

[$X \rightarrow$ They X$_1$ an X$_2$ $\bullet, 0, 1$]
[Sie haben eine Erhöhung vorausgesagt, 0.15, $(6, 2, 4)$]
[P_{LM}(Sie haben eine Erhöhung vorausgesagt)]
[$Mark_m$(Sie haben eine Erhöhung vorausgesagt)]
[$X \rightarrow$ They X$_1$ an X$_2$ $\bullet, 0, 1$]
[Sie prognostizierten eine Erhöhung, 0.02, $(5, 1, 4)$]
[P_{LM}(Sie prognostizierten eine Erhöhung)]</td></tr>
<tr><td></td><td>[$X \rightarrow$ predicted$\bullet, 1, 2$]
[sind ; von ; ausgegangen, $0.6, 3$]
[P_{LM}(sind) $*$ P_{LM}(von) $*$ P_{LM}(ausgegangen)]</td><td></td><td>[$X \rightarrow$ increase$\bullet, 3, 4$]
[Erhöhung, $1.0, 4$]
[P_{LM}(Erhöhung)]</td></tr>
<tr><td></td><td>[$X \rightarrow$ predicted$\bullet, 1, 2$]
[haben ; vorausgesagt, $0.3, 2$]
[P_{LM}(haben) $*$ P_{LM}(vorausgesagt)]</td><td></td><td></td></tr>
<tr><td></td><td>[$X \rightarrow$ predicted$\bullet, 1, 2$]
[prognostizierten, $0.1, 1$]
[P_{LM}(prognostizierten)]</td><td></td><td></td></tr>
<tr><td>They</td><td>predicted</td><td>an</td><td>increase</td></tr>
<tr><td>0 1</td><td>1 2</td><td>2 3</td><td>3 4</td></tr>
</table>

Figure 5.20: Language model scoring of translation options.

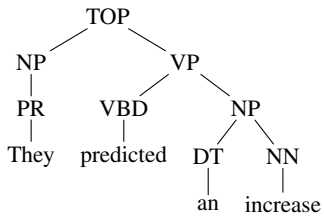


Figure 5.21: English sentence E with parse tree

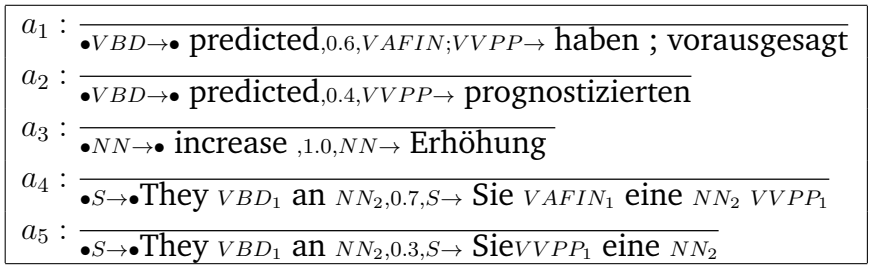


Figure 5.22: Axioms created with syntactically annotated Sh-l-MBOT rules.

<table>
<tr><td colspan="4">$[S\bullet \rightarrow$ They VBD_1 an $NN_2\bullet, 0, 4]$
$[S \rightarrow$ Sie prognostizierten eine Erhöhung$, 0.12, 11, (9, 3)]$

$[\bullet S \rightarrow$ They VBD_1 an $NN_2\bullet, 0, 4]$
$[S \rightarrow$ Sie haben eine Erhöhung vorausgesagt$, 0.42, 10, (8, 3)]$</td></tr>
<tr><td colspan="3">$[\bullet S \rightarrow$ They VBD_1 an $\bullet\ NN_2, 0, 3]$
$[S \rightarrow$ Sie $VVPP_1$ eine $NN_2, 0.12, 9, (7)]$

$[\bullet S \rightarrow$ They VBD_1 an $\bullet\ NN_2, 0, 3]$
$[S \rightarrow$ Sie $VAFIN_1$ eine $NN_2\ VVPP_1, 0.42, 8, (6)]$</td><td></td></tr>
<tr><td colspan="2">$[\bullet S \rightarrow$ They $VBD_1\bullet$ an $NN_2, 0, 2]$
$[S \rightarrow$ Sie $VVPP_1$ eine $NN_2, 0.12, 7, (5, 2)]$

$[\bullet S \rightarrow$ They $VBD_1\bullet$ an $NN_2, 0, 2]$
$[S \rightarrow$ Sie $VAFIN_1$ eine $NN_2\ VVPP_1, 0.42, 6, (4, 1)]$</td><td></td><td></td></tr>
<tr><td>$[\bullet S \rightarrow$ They $\bullet\ VBD_1$ an $NN_2, 0, 1]$
$[S \rightarrow$ Sie $VVPP_1$ eine $NN_2, 0.3, 5]$

$[\bullet S \rightarrow$ They $\bullet\ VBD_1$ an $NN_2, 0, 1]$
$[S \rightarrow$ Sie $VAFIN_1$ eine $NN_2\ VVPP_1, 0.7, 4]$</td><td>$[VBD\bullet \rightarrow$ predicted$\bullet, 1, 2]$
$[VVPP \rightarrow$ prognostizierten$, 0.4, 2]$

$[VBD\bullet \rightarrow$ predicted$\bullet, 1, 2]$
$[VAFIN; VVPP \rightarrow$ haben ; vorausgesagt$, 0.6, 1]$</td><td></td><td>$[NN\bullet \rightarrow$ increase$\bullet, 3, 4]$
$[NN \rightarrow$ Erhöhung$, 1.0, 3]$</td></tr>
<tr><td>They</td><td>predicted</td><td>an</td><td>increase</td></tr>
<tr><td>0 1</td><td>1 2</td><td>2 3</td><td>3 4</td></tr>
</table>

Figure 5.23: Chart parsing with annotated Sh-l-MBOT rules.

Pruning and language model scoring are done in exactly the same way as for non-annotated Sh-l-MBOT grammars.

5.4.3 Target Syntactic Annotations

Decoding with Sh-l-MBOT grammars annotated on the target language side can be done by simply removing the source non-terminal labels from the axioms and inference rules given in Equations 5.13, 5.15 and 5.16. The inference rules obtained in this way are given in Equations 5.17, 5.18 and 5.19, where the labeled source non-terminals are simply replaced by X. As in this configuration there is no need for the source left-hand-side of the rules to match any input parse tree, the inference rule in Equation 5.14 is dropped. Only the source right-hand sides of the rules are dotted.

$$[X \rightarrow \alpha\bullet, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w] \tag{5.17}$$

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j, w]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, B_1, \ldots, B_n \rightarrow \beta_1 \ldots \beta_n, i, j+1, w]} \tag{5.18}$$

$$\frac{\begin{array}{c}[X \rightarrow \alpha_1 \bullet X_m\alpha_2, B_1, \ldots, B_n \rightarrow \beta_{11}D_{1m}\beta_{12} \ldots \beta_{l1}D_{lm}\beta_{l2}, i, j, w_1] \\ [X \rightarrow \gamma\bullet, D_1 \ldots D_l \rightarrow \delta_1 \ldots \delta_l, j, k, w_2]\end{array}}{[X \rightarrow \alpha_1 X_m \bullet \alpha_2, B_1, \ldots, B_n \rightarrow \beta_{11}\delta_1\beta_{12} \ldots \beta_{l1}\delta_l\beta_{l2}, i, k, w_1 * w_2]} \tag{5.19}$$

With this new parse items, the decoding procedure works in exactly the same way as for Sh-l-MBOT grammars annotated on source and target language sides.

5.5 Evaluation of Shallow l-MBOT

In [Seemann et al., 2015b], a systematic evaluation of SMT systems with Sh-l-MBOT grammars is given. We present this result here. This evaluation measures translation quality for three types of Sh-l-MBOT grammars:

1. Without syntactic annotations
2. With annotations on source and target language side
3. With annotations on target language side only

All systems are evaluated on three language pairs:

1. English-to-German
2. English-to-Arabic
3. English-to-Chinese

We begin by describing the linguistic resources used for the experiments.

5.5.1 Linguistic Resources

The data for the English-to-German translation task is given in Table 5.1. [Seemann et al., 2015b] parsed the German side of the training data using BitPar [Schmid, 2004] without the morphological annotations. We used the Berkeley Parser [Petrov et al., 2006] to prepare the English side of the training data.

The data for the English-to-Arabic translation task is given in Table 5.2. Both sides of the training data have been parsed using the Berkeley Parser [Petrov et al., 2006]. In addition, we used MADA [Habash et al., 2009] to tokenize the Arabic data. Transliteration was done according to [Buckwalter, 2002].

The data for the English-to-Chinese translation task is given in Table 5.3. The Berkeley Parser [Petrov et al., 2006] was used to prepare both sides of the training data.

For all data sets, we performed length-ratio filtering. Furthermore, we word

Resources for English-to-German translation task	
training data	7th EuroParl corpus [Koehn, 2005]
training data size	$\approx$ 1.8M sentence pairs
language model	5-gram SRILM [Stolcke, 2002]
add. LM data	WMT 2013 [Bojar et al., 2013]
LM data size	$\approx$ 57M sentences
tuning data	WMT 2013
tuning size	3,000 sentences
test data	WMT 2013
test size	3,000 sentences

Table 5.1: Summary of resources for the English-to-German task.

Resources for English-to-Arabic translation task	
training data	MultiUN corpus [Eisele and Chen, 2010]
training data size	$\approx$ 5.7M sentence pairs
language model	5-gram SRILM [Stolcke, 2002]
add. LM data	Arabic in MultiUN
LM data size	$\approx$ 9.7M sentences
tuning data	held out from MultiUN
tuning size	2,000 sentences
test data	held out from MultiUN
test size	1,000 sentences

Table 5.2: Summary of resources for the English-to-Arabic task.

aligned all training sets using GIZA++ [Och and Ney, 2003] together with the *grow-diag-final-and* heuristic [Koehn et al., 2005].

5.5.2 Results

[Seemann et al., 2015b] evaluate the Sh-l-MBOT systems 1-3 enumerated above (Section 5.5) against the corresponding components of the Moses open source toolkit [Hoang et al., 2009]. The toolkit implements hierarchical rules extracted following the procedure in [Chiang, 2005] as well the decorated rules described in Section 4.3. These systems serve as baseline and their Sh-l-MBOT variants are the contrastive systems. We call configurations without syntactic annotations *Hier-*

Resources for English-to-Chinese translation task	
training data	MultiUN corpus [Eisele and Chen, 2010]
training data size	$\approx$ 1.9M sentence pairs
language model	5-gram SRILM [Stolcke, 2002]
add. LM data	Chinese in MultiUN
LM data size	$\approx$ 9.5M sentences
tuning data	NIST 2002, 2003, 2005
tuning size	2,879 sentences
test data	NIST 2008 [NIST, 2010]
test size	1,859 sentences

Table 5.3: Summary of resources for the English-to-Chinese task.

archical, systems with source and target side syntactic annotations *Tree-to-tree* and systems with annotations only on the target side *String-to-tree*. For tree-to-tree systems, we compare systems using the minimal rules presented in Section 5.1.2 and non-minimal ones (Section 5.1.2). Sh-l-MBOT grammars have been extracted according to [Maletti, 2011] and [Seemann et al., 2015a]. The decoding procedures used are the ones that have been discussed in the present chapter (in Sections 5.3.1 and 5.4). The procedure for tree-to-tree decoding has also been presented in [Braune et al., 2013]. To better put Sh-l-MBOT systems in the context of SMT systems, we also present the results obtained by high-ranked systems on public shared tasks [Bojar et al., 2014] such as phrase-based systems [Koehn et al., 2003] or string-to-tree systems obtained following [Galley et al., 2004, Galley et al., 2006].

Translation quality is measured in terms of BLEU [Papineni et al., 2002] and significance testing is done according to [Koehn, 2004]. The results obtained by each system are reported in Tables 5.5, 5.6 and 5.7. For hierarchical Sh-l-MBOT systems, rule extraction did not scale to the size of the training data for the English-to-Arabic task. So we omit the evaluation of hierarchical Sh-l-MBOT systems on this language pair.

We begin by presenting the results of hierarchical systems, then we discuss tree-to-tree systems before concluding with string-to-tree systems.

5.5.3 Hierarchical systems

On all language pairs, hierarchical systems yield the best translation quality or at least comparable to the best performing system (on the English-German task). In the hierarchical setup, translation quality achieved with Sh-l-MBOT systems is comparable to the hierarchical system on the English-Chinese task and slightly worse on the English-German task. This lack of improvement is due to the fact that rules without syntactic annotations are very flexible and thus achieve high coverage. In such a scenario, the additional flexibility provided by target side discontiguities is limited. A detailed analysis of rule applications in hierarchical Sh-l-MBOT systems in [Seemann et al., 2015b] shows that in this setup Sh-l-MBOT rules are barely used by the SMT system.

5.5.4 Tree-to-tree systems

For both, the baseline and the contrastive systems, the tree-to-tree configuration performs worse than any other configuration. For the baseline, i.e. a system built on Synchronous Context Free Grammars (SCFG), our results confirm previous evaluations, such as [Ambati and Lavie, 2008] which we discuss in Section 2.2.1. Our evaluation shows that for Sh-l-MBOT systems, the same result holds on all language pairs. Moreover, the results show that making rules more flexible by allowing sequences of trees on the target language side hurts performance rather than yielding improvements. The results obtained by systems with minimal Sh-l-MBOT rules are particularly bad. By substituting minimal rules with non-minimal ones, large improvements are obtained. These are, however, not sufficient to beat the tree-to-tree baseline. This result is quite opposed to the results presented in [Sun et al., 2009] which report that STSSG yield very large improvements for a Chinese-English translation task (see Section 2.2.3). This divergence is probably due to the fact that our system is quite different from [Sun et al., 2009] as has been shown in Section 2.3. Another reason is that we evaluate on English-to-Chinese.

System	**BLEU**
Moses tree-to-tree Baseline	12.60
sh-ℓMBOT	*13.06

Table 5.4: Evaluation of a system combining SCFG and *sh*-ℓMBOT rules. The starred results are statistically significant improvements over the baseline (at confidence $p < 0.05$).

In [Braune et al., 2013], we have shown that systems where minimal Sh-l-MBOT rules are combined with SCFG rules yield modest but significant improvements over a tree-to-tree baseline. The compared systems have been evaluated on the news translation task of WMT 2009 [Callison-Burch et al., 2009]. The training data for both systems is from the 4th version of the *Europarl* [Koehn, 2005] and the *News Commentary* corpus. The English side of the training data was parsed with the Charniak parser of [Charniak and Johnson, 2005], and the German side with BitPar [Schmid, 2004] without function and morphological annotations. Our German language model (4-gram) was trained on the German side of the training data augmented by the Stuttgart SdeWaC corpus [Web-as-Corpus Consortium, 2008], detailed in [Baroni et al., 2009]. The result is given in Table 5.4 where Sh-l-MBOT denotes our system using a tree-to-tree SCFG grammar combined with a tree-to-tree Sh-l-MBOT grammar.

5.5.5 String-to-tree systems

An extensive evaluation of string-to-tree systems has been presented in [Seemann et al., 2015a]. In this section we discuss these systems in terms of translation quality. The results for the baseline and contrastive systems show that on all language pairs, string-to-tree systems achieve better quality than tree-to-tree configurations. The results show improvements up to 5 BLEU points on the English-Arabic language pair. For SCFG based systems, this result confirms previous work ([Ambati and Lavie, 2008],[Williams and Koehn, 2012],[Galley et al., 2004]). We show that for Sh-l-MBOT systems a similar improvement is observed. A very in-

Results of English-to-German translation task		
Setting	**System**	**BLEU**
tree-to-tree	Moses tree-to-tree Baseline	14.50
	minimal tree-to-tree *sh*-ℓMBOT	14.09
	non-minimal tree-to-tree *sh*-ℓMBOT	14.41
string-to-tree	Moses string-to-tree Baseline	14.96
	non-minimal string-to-tree *sh*-ℓMBOT	*15.49
	GHKM string-to-tree	17.10
hierarchical	Moses hierarchical Baseline	17.00
	hierarchical *sh*-ℓMBOT	16.57
	phrase-based Moses	16.80

Table 5.5: Evaluation results for the English-to-German translation task. The starred results are statistically significant improvements over the baseline (at confidence $p < 1\%$).

teresting result in the evaluation of string-to-tree systems is that in this configuration, Sh-l-MBOT systems significantly beat the Moses string-to-tree baseline on all language pairs. This shows that when (i) syntactic constraints are removed on the source language side of Sh-l-MBOT rules and (ii) there is no need for the rules to match an input parse tree, then having target side discontiguities improves translation quality.

The GHKM string-to-tree system implemented in Moses, which we discussed in Section 2.2.1 has been a high-ranked system on public shared tasks [Bojar et al., 2014]. Our results show that this system significantly outperforms other types of string-to-tree systems on the English-German task. However, for the English-Arabic language pair, it achieves the worst performance, while on English-Chinese it is comparable to Sh-l-MBOT.

5.6 Conclusion

We have presented a SMT system based on Sh-l-MBOT, a synchronous grammar which allows sequences of trees on the target side of the rules (see Section 2.1.3).

Results of English-to-Arabic translation task		
Setting	**System**	**BLEU**
tree-to-tree	Moses tree-to-tree Baseline	43.49
	minimal tree-to-tree *sh*-ℓMBOT	32.88
	non-minimal tree-to-tree *sh*-ℓMBOT	41.37
string-to-tree	Moses string-to-tree Baseline	48.23
	non-minimal string-to-tree *sh*-ℓMBOT	*49.10
	GHKM string-to-tree	46.66
	phrase-based Moses	50.71
	hierarchical phrase-based Moses	51.90

Table 5.6: Evaluation results for the English-to-Arabic translation task. The starred results are statistically significant improvements over the baseline (at confidence $p < 1\%$).

After a brief introduction of Sh-l-MBOT rule extraction, we have presented the translation model and decoding procedures of our Sh-l-MBOT system. Our approach can deal with rules without syntactic annotations or with annotations at different levels such as, for instance, tree-to-tree or string-to-tree rules. An extensive evaluation of SMT with Sh-l-MBOT, given in Section 5.5, has shown that string-to-tree Sh-l-MBOT significantly outperform SCFG-based string-to-tree systems on several language pairs. In all other configurations (such as tree-to-tree) Sh-l-MBOT achieves performance that is either worse (tree-to-tree) or comparable (hierarchical) to the corresponding SCFG-based baselines. However, we have also shown that tree-to-tree systems combining Sh-l-MBOT rules with SCFG outperform SCFG-based tree-to-tree systems.

The major shortcoming of this work is the computation of the language model scores. Our scoring function assumes that the elements of discontiguous translation options are independent although it is known in advance that they will eventually be combined. Interesting future work on this topic would be to use a dependency-based language model to score discontiguous translation options.

Results of English-to-Chinese translation task		
Setting	**System**	**BLEU**
tree-to-tree	Moses tree-to-tree Baseline	17.63
	minimal tree-to-tree *sh*-ℓMBOT	12.01
	non-minimal tree-to-tree *sh*-ℓMBOT	16.77
string-to-tree	Moses string-to-tree Baseline	17.69
	non-minimal string-to-tree *sh*-ℓMBOT	*18.35
	GHKM string-to-tree	18.33
hierarchical	Moses hierarchical Baseline	18.74
	hierarchical *sh*-ℓMBOT	18.60
	phrase-based Moses	18.09

Table 5.7: Evaluation results for the English-to-Chinese translation task. The starred results are statistically significant improvements over the baseline (at confidence $p < 1\%$).

Chapter 6

Improved Rule Selection for Hierarchical Machine Translation

We present our first contribution to the topic of syntax-based SMT with soft syntactic constraints: the definition of a global rule selection model and its integration into a hierarchical system. Hierarchical SMT has been introduced in Section 4.1. A detailed overview of this contribution is given in Section 3.4.

We begin by presenting the general idea behind our rule selection model in Section 6.1. This overall presentation can also be found in [Tamchyna et al., 2014] who describe the integration of a discriminative classifier in a phrase-based as well as a hierarchical system. Then we present the definition of our model as well as the used training procedure in Sections 6.2 and 6.3. The integration of the model in the hierarchical decoding process is presented in Section 6.4. Finally, we discuss the advantages of our model over previous work in Section 6.5, which we confirm empirically in our evaluation (Section 6.6). We conclude this chapter by explaining shortcomings of our approach and possible future work.

6.1 Overall Presentation

The goal of our model is to improve the hierarchical rule application process presented in Section 3.1 with a global syntactic rule selection model. Each time a rule is applied to translate a given span in the input sentence, we check whether or not this rule is the only candidate for this source segment. If there are multiple candidate rules (i.e with the same source side) we query a model that resolves the ambiguity between these rules by using richer information than relative frequencies in the training data or any other feature of the hierarchical translation model (in Section 4.1.2). By "resolving the ambiguity", we denote the assignment of a score to the considered rule. This score is drawn from a probability distribution over all candidate rules.

To illustrate our approach, consider rules q_1 and q_2 introduced in Section 3.1 as well as Sentence F_1. Suppose that the weights of the rules are the direct translation probabilities in Section 4.1.2. Rule q_2 has been observed more frequently in the training data and hence has a weight of 0.4 compared to 0.2 for q_1.

q_1 $X \xrightarrow{0.2} \langle\ X_1$ pratique X_2, practical $X_1\ X_2\ \rangle$
q_2 $X \xrightarrow{0.4} \langle\ X_1$ pratique X_2, $X_1\ X_2$ process $\rangle$

F_1 Une étude de l' (intérêt)$_{X_1}$ **pratique** (de notre approche)$_{X_2}$.
A study on the (interest)$_{X_1}$ ***practical*** *(of our approach)$_{X_2}$.*

During decoding, rules q_1 and q_2 are candidate rules to apply on the segment *intérêt pratique de notre approche*. Instead of relying on the weights of the rules, which would lead to the erroneous application of q_2, our system queries a rule selection model M that integrates rich information (i.e. more than relative frequencies in training), such as the structure of the source sentence or the shape of the candidate rules. For our illustration, we suppose that M uses the pieces of information I_1 to I_4 given in Figure 6.1.

Id	Description	Rule q_1	Rule q_2
I_1	Relative position of non-terminals	straight	straight
I_2	Pair of terminals in rule	pratique-practical	pratique-process
I_3	Syntactic structure of application segment	Incomplete	
I_4	POS to the left of application segment	Determiner	

Figure 6.1: Information used by model M

In order to query M, the pieces of information I_1 to I_4 must be provided. Consequently, our system extracts this information from the required sources, i.e. sentence F_1 and its syntactic structure, given in Figure 8.2.2, as well as q_1 and q_2. The query then returns the probability of a candidate rule to be correct given the provided information. In our example, we suppose that the used information allows a good disambiguation between q_1 and q_2 and hence returns the probabilities 0.8 for q_1 and 0.2 for q_2.

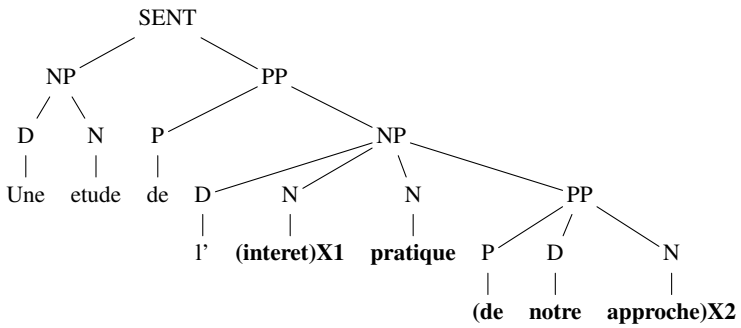


Figure 6.2: Parse tree of Sentence F_1

In the next sections, we first discuss how to define and train M (Section 6.2) and how to integrate it in a hierarchical decoder to obtain the desired predictions (Section 6.4) during rule application.

6.2 Rule Selection Model

The disambiguation function of the rule selection model M can be formulated as the task of choosing the correct target side of a hierarchical rule given knowledge

about its source side as well as additional pieces of information such as I_1 to I_4. This task can be modeled as a multi-class classification problem where each target-side corresponding to a source side gets a label. The pieces of information taken into account by M are the features of the model. A first set of features for our model includes the shape of hierarchical rules. Examples of such features are I_1 and I_2 above. A second set contains information about the structure of the source sentence such as I_3 and I_4 above.

6.2.1 Model Definition

In order to define a rule selection model, a formal representation of hierarchical rules is needed. We use the same notation as in Section 4.1.1 and denote hierarchical rules by $X \rightarrow \langle \alpha, \gamma, \sim \rangle$, where α and γ are the source and target language strings and $\sim$ the correspondence between non-terminal symbols. In a second step, we define representations for our features. The features on the shape of hierarchical rules are denoted by $R(\alpha, \gamma)$. The features on the source sentence f are denoted by $C(f, \alpha)$. Our rule selection model estimates $P(\gamma \mid \ \alpha, C(f, \alpha), R(\alpha, \gamma))$, which is the probability to see the target side of a rule given its source side and the features presented above. This distribution can be written as:

$$P(\gamma \mid \alpha, C(f, \alpha), R(\alpha, \gamma)) = \frac{\prod_i h_i(\alpha, C(f, \alpha), R(\alpha, \gamma))^{\lambda_i}}{\sum_{\gamma' \in GTO(\alpha)} \prod_i h_i(\alpha, C(f, \alpha), R(\alpha, \gamma'))^{\lambda_i}} \tag{6.1}$$

The model score is normalized over the set G' of candidate target sides γ' for a given α. The function $GTO : \alpha \rightarrow G'$ generates, given the source side, the set G' of all corresponding target sides γ'.

As seen in Section 3.2, rule selection models proposed in the literature are of two types. Local models [Chan et al., 2007, He et al., 2008, He et al., 2010] train one classifier for each source side of a rule. In contrast, global models [Cui et al., 2010] train a single classifier on all rules. Our model is global.

6.2.2 Feature templates

A seen in the previous section, the features of our model are of the following types:

1. The rule shape features $R(\alpha, \gamma)$ model the structure of hierarchical rules. These features are similar to [Setiawan et al., 2009, Cui et al., 2010, Simianer et al., 2012] (see Section 3.2).
2. The source context features $C(f, \alpha)$ model the structure of the input sentence. In the same fashion as [Marton and Resnik, 2008, Cui et al., 2010] our contextual features include soft syntactic constraints. Additionally, we combine these features with a set of lexical features. For comparison to approaches that only work with lexical features, such as [Chan et al., 2007, He et al., 2008, He et al., 2010] (see Section 3.2), we also train models using lexical features only.

The rule shape features of our model are given in Figure 6.3. They capture information about the source side α and target side γ of hierarchical rules. Examples are the words in α and γ or the alignment between the non-terminals in a rule. To illustrate these features, suppose that rule shape features are extracted from rule q_2 above. These features are given in the rightmost column of Figure 6.3. For instance, the words in the source side of q_2 are X_1 *pratique* X_2. The alignment between non-terminals is $X_1 \leftrightarrow X_1$ $X_2 \leftrightarrow X_2$.

The syntactic source context features are given in Figure 6.4. They capture information about the syntactic structure of the source sentence. More precisely, they describe the position of the application span in the parse tree of the source sentence. They indicate, for instance, if a rule is located in a complete or incomplete syntactic constituent. If the rule is in a complete syntactic constituent, the type of constituent is reported as well as its parent. If it is in an incomplete constituent, it is labeled with NOTAG and its lowest parent is indicated. Another feature is the application span width in the input parse tree. To illustrate the syntactic source context features, suppose that syntactic features are extracted from the parse tree of sentence F_1 given in Figure 6.2. The candidate rules are q_1 or q_2. These features

Feature Template	Example
Source side α	$X1$*_pratique_*$X2$ (one feature)
Words in α	*X1 pratique X2* (three features)
Target side γ	*X1 X2 process*
Words in γ	*X1 X2 process*
Aligned terminals in α and γ	*pratique*↔*process*
Aligned non-terminals in α and γ	*X1*↔*X1 X2*↔*X2* (two features)
Best baseline translation probability	*Most_Frequent*

Figure 6.3: Rule shape features

Feature Template	Example
Does α match a constituent	*no_match*
Type of matched constituent	NOTAG
Parent of matched constituent	NOTAG
Lowest parent of unmatched constituent	*NP*
Span width covered by α	*5*

Figure 6.4: Syntactic features

are given in the rightmost column in Figure 6.4. For instance the applied rule does not match any syntactic constituent and its lowest parent is an NP.

The lexical source context features are given in Figure 6.5. They capture information about the lexical structure of the sentence to be translated. Examples are the enriched words to the right and left of the application span of a rule. Each enriched word is composed of a surface form, POS tag and lemma. To illustrate these features, consider that lexical features are extracted from sentence F_1 to which q_1 or q_2 is applied. These features are given in the rightmost column of Figure 6.5. For instance, the words to the right and left of the rule application are the determiner *l'* and the punctuation mark *column (.)*.

Feature Template	Example
first factored form left of α	*la, D, la*
second factored form left of α	*de, P, de*
first factored form right of α	*., PONCT, .*
second factored form right of α	*None, None, None*

Figure 6.5: Lexical features

6.3 Model training

As seen above, the goal of our rule selection model is to choose the correct target language side of a hierarchical rule given its source side. Learning such a model requires training data that indicates, for each possible source side of a hierarchical rule which corresponding target language sides are correct[1] and which ones are not. Moreover the training data needs to indicate under which features a given target language side is correct or incorrect. Such training data can easily be created by using the hierarchical rule extraction heuristics [Chiang, 2007] presented in Section 4.1.1.

6.3.1 Creation of Training Examples

To create the training examples for our model, we first use the input to the hierarchical rule extraction, i.e, a word aligned parallel corpus such as the one in Figure 4.1, Section 4.1.1. Each time a rule r can be extracted according to the heuristics in [Chiang, 2007], we create a new training example. The set of rule shape features for this example are extracted from the source and target sides α and γ of r. The set of source context features are obtained by looking at the source sentence S from which the rule has been extracted. The target language side γ of r is the right one in the context of S. Consequently, it is a correct class for this example and gets a cost of 0. The incorrect classes are all rules q_i that have the same source side as

[1] Due to unaligned words, several target sides can be correct given a source side.

r but a different target side. These can be obtained by consulting the hierarchical grammar (HG) extracted from the considered parallel corpus and collecting all rules that have the same source side as r. All target sides that are different from γ are incorrect classes and get a cost of 1.

To illustrate this procedure, consider (i) that rule q_1 has been extracted from the parallel sentences F_2 and E_2, (ii) that the HG contains rules q_2 to q_6 which have the same source side as q_1, and (iii) rules get a label corresponding to the number of their entry in the rule table.[2]

F_2 Les avantages de l' (aspect)$_{X_1}$ **pratique** (de la robotique)$_{X_2}$.
E_2 The advantages of the **practical** (aspect)$_{X_1}$ (of robotics)$_{X_2}$.

q_1 $X \xrightarrow{0.2} \langle\ X_1$ pratique X_2, practical $X_1\ X_2\ \rangle$
q_2 $X \xrightarrow{0.4} \langle\ X_1$ pratique X_2, $X_1\ X_2$ process $\rangle$
q_3 $X \xrightarrow{0.1} \langle\ X_1$ pratique X_2, X_1 practical $X_2\ \rangle$
q_4 $X \xrightarrow{0.15} \langle\ X_1$ pratique X_2, X_1 helpful $X_2\ \rangle$
q_5 $X \xrightarrow{0.05} \langle\ X_1$ pratique X_2, $X_1\ X_2$ practice $\rangle$
q_6 $X \xrightarrow{0.1} \langle\ X_1$ pratique X_2, practice $X_2\ X_1\ \rangle$

The created training example is given in Figure 6.6. The example contains features about the source and target side of q_1 such as the rule shape features given in the rightmost column of Figure 6.6. The lexical and syntactic source context features are the same as in Figures 6.5 and 6.4 (Section 6.2.2). The target side of q_1 given in the first row is the correct class and gets a cost of 0. The target sides of rules q_2 to q_6 are incorrect classes and get a cost of 1.

The procedure above leads to a huge number of training examples, which we present in Section 6.5.2. For each training example there are many incorrect classes. We reduce these by applying significance testing [Johnson et al., 2007] on the HG used to collect the incorrect classes. Aside of that, we do not perform any further pruning of negative instances (incorrect classes) in contrast to [Cui et al., 2010] (see Section 3.2). As a consequence, we can only train our model with a restricted

[2]For instance, the 10th rule in the rule-table gets the label 10.

Target Side	Label	Cost	Rule Shape features of q_1	Values
practical X_1 X_2	10	0	Source side α	*X1_pratique_X2*
X_1 X_2 process	15	1	Words in α	*X1 pratique X2*
X_1 practical X_2	16	1	Target side γ	*practical_X1_X2*
X_1 helpful X_2	17	1	Words in γ	*practical X1 X2*
X_1 X_2 practice	18	1	Aligned terminals	*pratique↔practical*
practice X_2 X_1	19	1	Aligned non-terminals	*X1↔X1 X2↔X2*

Figure 6.6: Training Example for rule q_1 extracted from F_2, E_2 with rule shape features.

amount of training data. Note that the global model in [Cui et al., 2010] uses the same amount of training data and still needs to prune out negative samples.

6.3.2 Training Algorithm

To train our model, we use the high-speed classifier Vowpal Wabbit[3] (VW) with the cost-sensitive one-against-all-reduction [Beygelzimer et al., 2005]. More precisely, this is the label dependent version of Cost -Sensitive One-Against-All which uses classification.[4]

6.4 Decoding with Improved Rule Selection

During decoding, our rule selection model is queried to disambiguate between candidate rules. In terms of the hierarchical search procedure in Section 4.2.1, this means that each time a new item is proved for a span $[i, j]$, the rule selection model is queried to get an additional weight w' for this item. As the weight of the parse items is in fact a combination of the features in the translation model, our model prediction is an additional feature taken into account to compute w.

In order to get the model prediction, the decoder must provide the features defined in the rule selection model. This requires to have access to (i) the source

[3] `http://hunch.net/~vw/`. Implemented by John Langford and many others.
[4] The command line parameter to VW is "csoaa_ldf mc".

and target side of the rule in the parse item as well as (ii) the source sentence that is being processed with lemma and POS tags as well as syntactic structure. To access the source *and* target side (γ) of the rule during decoding, the parsing procedure in Section 4.2.1 needs to be adjusted to integrate γ. As seen in Section 4.2, γ is dropped when non-terminal symbols are processed and is replaced by a partial translation which is a combination $\gamma_1 \otimes \gamma_2$ of two target sides. The operator $\otimes$ denotes the replacement of non-terminal symbols with the target sides of passive items. Second, the source sentence must be accessed during decoding and hence be part of the parse items. We integrate source sentence information in the items through a function that queries the model for prediction and updates the weight of the item. This function replaces the weight of each item.

6.4.1 Adjustments to the CYK+ Parsing Algorithm

To include a rule selection model in the hierarchical decoding process, we have to include the target side of the rules into the parse items. Each item carries the original target side Γ of a hierarchical rule in addition to its source side and partial translation. When new items are created using lexical rules, antecedent and consequent carry Γ. When non-lexical rules are applied, the consequent carries the target side Γ_1 of the item that is active during inference. Consequently, Equations 4.9 and 4.10 in Section 4.2.2 are rewritten as:

$$\frac{[X \rightarrow \alpha_1 \bullet t_{j+1}\alpha_2, i, j, w, \gamma, \Gamma]}{[X \rightarrow \alpha_1 t_{j+1} \bullet \alpha_2, i, j+1, w, \gamma, \Gamma]} \tag{6.2}$$

$$\frac{[X \rightarrow \alpha_1 \bullet X\alpha_2, i, j, w_1, \gamma_1, \Gamma_1] \quad [X \rightarrow \beta\bullet, j, k, w_2, \gamma_2, \Gamma_2]}{[X \rightarrow \alpha_1 X \bullet \alpha_2, i, k, w_1 * w_2, \gamma_1 \otimes \gamma_2, \Gamma_1]} \tag{6.3}$$

The adjusted parse items in Equations 6.2 and 6.3 must further be modified to include the access to the source sentence and the integration of model predictions. This is done by replacing the weight w of each parse item with the function

$f([i,j], S[0,\ldots,i-1,j+1,\ldots,|s|], \alpha, \gamma, h, w)$ where s is the source sentence. We simply denote our function by $F()$. Its arguments are:

1. The span $[i,j]$ of the item.
2. The segment $S[0,\ldots,i-1,j+1,\ldots,|s|]$ of the source sentence around $[i,j]$.
3. The source and target sides α and γ of the rule.
4. The weight w of the parse item without the prediction of the rule selection model.
5. The list $h[X]$ of translation options for span $[i,j]$.

The first three arguments are used to compute the model features. The fourth argument is the current weight of the item, which is updated by $F()$. The first three arguments are easily accessible during decoding. The last argument is the list $h[X]$ containing the translation options spanning from i to j with the same source side α. This list is required to transform the output of the classifier, i.e. the numerator of Equation 6.1, into a probability distribution over all rules. In fact, the sublist $h[X]$ is the output value of the function $GTO : \alpha \rightarrow G'$ in Equation 6.1. In contrast to standard weight computation[5], this normalization can only take place when all translation options have been collected. This means that the function $F()$ can only be evaluated when the list $h[X]$ is complete. To allow this, the standard CYK+ search procedure in Algorithm 1 must be adjusted. The adjustment consists in replacing lines 9 and 10 with lines 1 to 7 below.

Algorithm 5 CYK+ search algorithm with evaluation of $F()$

1: **for all** items $[X \rightarrow \alpha\bullet, i, j, w, \gamma, \Gamma]$ provable from items in $a[X]$ and $d[X]$ and $h[X]$ **do**
2: Add $[X \rightarrow \alpha\bullet, i, j, w, \gamma, \Gamma]$ to $h[X]$
3: **end for**
4: **for all** items $[X \rightarrow \alpha\bullet, i, j, w, \gamma, \Gamma]$ in $h[X]$ **do**
5: Compute $w' = F(w)$ // The value of w' can only be computed after having collected all translation options
6: Update weight w
7: **end for**

Note that we do not adjust glue-rule applications as glue rules have no competing

[5]Which takes place as soon as a new item is created.

rules.

6.4.2 Example

We illustrate our adjusted procedure by decoding the segment *l' intérêt pratique de notre approche* using the axioms b_1 to b_6 given in Table 6.7. These axioms carry rules q_1 to q_4 presented in the previous sections as well as r_1 and r_2 below. The translation options $h[X]$ created during decoding are given in Table 6.8. For the translation options spanning $[1, 2]$ and $[3, 6]$, we assume that $F()$ has already been computed. As there are no competing axioms for those spans, we assume that our model prediction is 1.0. For span $[1, 6]$, axioms b_1 to b_4 are competing because they carry rules with the same source side X_1 *pratique* X_2. For each of the translation options created using these axioms (together with the translation options of sub-spans $[1, 2]$ and $[3, 6]$), the weight $F_i()$ that includes our rule selection model has to be computed.

r_1 $X \xrightarrow{1.0}$ ⟨ intérêt, interest ⟩
r_2 $X \xrightarrow{1.0}$ ⟨ de notre approche, of our approach ⟩

b_1 : $\overline{X \rightarrow \bullet X_1 \text{ pratique } X_{2,0.2,}\text{practical } X_1 \ X_2}$	b_2 : $\overline{X \rightarrow \bullet X_1 \text{ pratique } X_{2,0.4,} X_1 \ X_2 \text{ process}}$
b_3 : $\overline{X \rightarrow \bullet X_1 \text{ pratique } X_{2,0.3,} X_1 \text{ practical } X_2}$	b_4 : $\overline{X \rightarrow \bullet X_1 \text{ pratique } X_{2,0.3,} X_1 \text{ helpful } X_2}$
b_5 : $\overline{X \rightarrow \bullet \text{intérêt}_{,1.0,}\text{interest}}$	b_6 : $\overline{X \rightarrow \bullet \text{de notre approche}_{,1.0,} \text{ of our approach}}$

Figure 6.7: Axioms created from rules q_1 to q_4 and r_1 to r_2

Table 6.4.2 shows the computation of the weight function w for each translation option (TO). The arguments of the function are (i) the source sentence segment surrounding the span of the TO (denoted by Source Sent.), (ii) the source side α in the TO, (iii) the target side Γ in the TO (iv) the weight of the TO and (v) the list of all TO for this span with the same α. Using these arguments we extract the features of the rule selection model presented in Section 6.2, query the model and multiply the obtained prediction to the weight of the translation option.

	$[X_1$ pratique $X_2\bullet, 1, 6]$ $[F_4,$ interest helpful of our approach, X_1 helpful $X_2, 6, (1, 2)]$ $[X_1$ pratique $X_2\bullet, 1, 6]$ $[F_3,$ interest practical of our approach, X_1 practical $X_2, 5, (1, 2)]$ $[X_1$ pratique $X_2\bullet, 1, 6]$ $[F_2,$ interest of our approach process, X_1X_2 process$, 4, (1, 2)]$ $[X_1$ pratique $X_2\bullet, 1, 6]$ $[F_1,$ practical interest of our approach, practical$X_1X_2, 3, (1, 2)]$					
	$[X \rightarrow$ intérêt$\bullet, 1, 2]$ $[1.0,$ interest, interest$, 1]$		$[X \rightarrow$ de notre approche$\bullet, 4, 6]$ $[1.0,$ of our approach, of our approach$, 2]$			
l’	intérêt	pratique	de	notre	approche	.
0 1	1 2	2 3	3 4	4 5	5 6	6 7

Figure 6.8: Rule application with rule selection model.

TO Id	Source Sent.	α	γ	weight	TO list
3	[l’ .]	X_1 pratique X_2	practicalX_1X_2	0.2	$\{3, 4, 5, 6\}$
4	[l’ .]	X_1 pratique X_2	X_1X_2 process	0.4	$\{3, 4, 5, 6\}$
5	[l’ .]	X_1 pratique X_2	X_1 practical X_2	0.3	$\{3, 4, 5, 6\}$
6	[l’ .]	X_1 pratique X_2	X_1 helpful X_2	0.3	$\{3, 4, 5, 6\}$

Figure 6.9: Arguments of the weight calculation for TO 3 to 6.

6.4.3 Integration into the Hierarchical Translation Model

Our model is integrated in the hierarchical translation model as an additional feature of hierarchical rules in Equation 4.3.

6.5 Advantages over Previous Rule Selection Models

As seen in Section 3.4, our contribution to hierarchical rule selection is the definition of a model that (i) is global and (ii) performs no pruning of negative instances (incorrect classes). In this section, we discuss why global models without pruning should perform better than local [Chan et al., 2007, He et al., 2008, He et al., 2010] or pruned [Cui et al., 2010] models. We will show that global mod-

els are better than local ones because they allow feature sharing (Section 6.5.1) and that aggressive pruning of negative instances hurts the quality of model predictions (Section 6.5.2). In our evaluation (Section 6.6) we show empirically that global models without pruning outperform their local or pruned counterparts.

6.5.1 Feature Sharing

Because they train one classifier for each set of hierarchical rules with the same source side, local models cannot share features between rules with different source sides. As opposed to these models, ours can perform feature sharing. To illustrate sharing, consider rule q_2, extracted from sentence F_2 (in Section 6.1). Also suppose that rules q_7 and q_8 below have been extracted from sentence F_3. The parse of F_3 is given in Figure 6.10.

F_3 Un article sur les **modèles** (statistiques)$_{X_1}$ de (bas niveau)$_{X_2}$
*A paper on the **models** (statistical)$_{X_1}$ of (low-level)$_{X_2}$*

q_2 X $\xrightarrow{0.4}$ ⟨ X_1 pratique X_2, X_1 X_2 process ⟩:
q_7 X $\xrightarrow{0.7}$ ⟨ modèles X_1 de bas X_2, X_1 X_2 models ⟩
q_8 X $\xrightarrow{0.3}$ ⟨ modèles X_1 X_2, X_1 X_2 models ⟩

Rules q_2, q_7 and q_8 have completely different source sides. However, they share many features such as:

1. the POS tags of the first and second words to the left of the segment where they are applied (which are P and D)
2. the syntactic structure of this segment (which is that (i) it is not a complete constituent and (ii) it has a NP as its lowest parent)
3. the rule span width (which is 5)

Local models consider (and thus weight) these features independently for q_2, q_7 and q_8 while a global model considers these rules together. In the presence of data

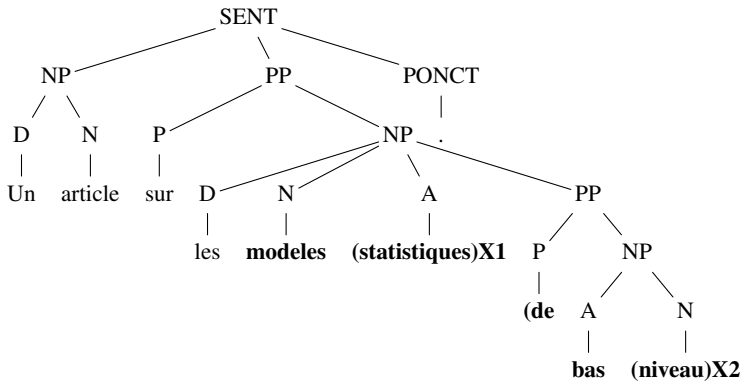


Figure 6.10: Parse tree of Sentence F_3

sparsity or lack of training data, the first option leads to a bad weighting of features (1)-(3). For instance, suppose that among rules with the same source side as q_2 only a few are correctly applied in presence of features (1) to (3). A local model will give very low weights to these features. If these features model a frequent linguistic phenomenon but have simply not been seen often in the presence of q_2 in the training data, then a local model misses an important generalization and cannot provide a good prediction for rule application in the presence of features (1) to (3).

6.5.2 Training without Pruning of Negative Examples

The procedure to create training instances for a global model leads to a very large collection of training examples, and for each of those a very large amount of incorrect classes, which we also call negative examples. As an illustration, Table 6.1 displays the total number of training examples for all data sets used in our evaluation (in Section 6.6).

In order to train a model with this amount of training data, a highly scalable classification framework must be used. As seen in Section 3.2, previous approaches using global models did not use such a framework and had to heavily prune out incorrect classes in the training examples in order not to ”choke the training procedure” [Cui et al., 2010, p.8]. As Vowpal Wabbit (VW) is extremely fast and supports

Data	Science	Medical	News
Sentences	139,215	111,165	150,000
Examples	47,952,867	25,435,958	40,062,073
cost 0	50,718,190	26,458,411	41,115,656
cost 1	493,271,397	170,064,556	446,105,435
avg 1	10.29	6.69	11.14

Table 6.1: Number of training examples. The last line shows the average number of negative samples (avg 1) for each training example.

online streaming we can train a global rule selection model without pruning out incorrect classes.

By avoiding pruning of negative examples, we keep important information for model training. As an illustration, consider rules q_2, q_3 and q_5. Suppose that a training example has been created from sentence F_1 (in Figure 6.2) where q_3 is a positive instance and q_2 and q_5 incorrect classes.

q_2 $X \xrightarrow{0.4} \langle\ X_1$ pratique X_2, $X_1\ X_2$ process $\rangle$: 0
q_3 $X \xrightarrow{0.1} \langle\ X_1$ pratique X_2, X_1 practical $X_2\ \rangle$: 1
q_5 $X \xrightarrow{0.05} \langle\ X_1$ pratique X_2, $X_1\ X_2$ practice $\rangle$: 0

The incorrect classes indicate that in the context of sentence F_1, the internal features of q_2 and q_5 are not correct. The information provided to the model by these classes can be paraphrased into I.

I In the syntactic and lexical context of F_1 the terminal *pratique* should neither be translated into *practice* nor into *process*

If rules q_2 and q_5 appear infrequently in the extracted hierarchical grammar, they are pruned out and information I is lost. This loss hurts the accuracy of the rule selection model. To illustrate this point, consider sentence F_4, which has a context similar to F_1 in terms of the lexical and syntactic features described in Section 6.2.2. The parse tree of F_4 is given in Figure 6.11.

F_4 Les avantages de l' (aspect)$_{X_1}$ **pratique** (de la robotique)$_{X_2}$.
The advantages of the (aspect)$_{X_1}$ ***practical*** *(of robotics)$_{X_2}$.*

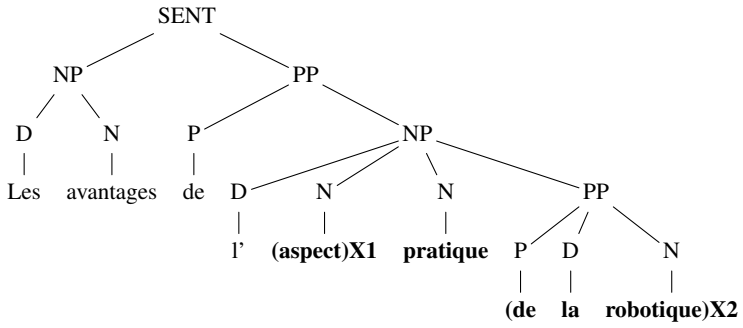


Figure 6.11: Parse tree of sentence F_4

If at decoding time competing rules sharing features with q_2 and q_5 are bad candidates to apply on F_1 and F_4 then the rule selection model cannot block their application based on information I. E.g, if rules q_6 and q_7 have high scores in the hierarchical model but are bad candidates to translate F_1 and F_4, then a pruned model fails to block their application: the discriminative model does not know that rules with lexical items *practice* and *process* on the target language side are bad candidates to translate F_1 and F_4. Consequently, the incorrect translations E_1^* and E_2^* might be created.

q_6 X $\rightarrow \langle$ X_1 pratique X_2, X_2 X_1 practice $\rangle$
q_7 X $\rightarrow \langle$ X_1 pratique X_2, X_1 X_2 process $\rangle$

E_1^* The advantages of the of robotics aspects practice
E_2^* The advantages of the aspects of robotics process

6.5.3 Feature combination

As seen above (Section 6.5.1), feature sharing between rules with different source sides improves the accuracy of rule selection models. Sharing can be done even more effectively when different feature templates are combined with each other.

As an example, consider rules q_2, q_7 and q_8 presented in Section 6.5.1 which share features (1) to (3):[6]

1. the POS tags of the first and second words to the left of the segment where they are applied (which are P and D)
2. the syntactic structure of this segment (which is that (i) it is not a complete constituent and (ii) it has a NP as its lowest parent)
3. the rule span width (which is 5)

By combining these features with each other, new templates can be generated, which allow to share more complex features. For instance by combining features (1),(2) and (3), a complex pattern can be generated, which indicates that in addition to sharing features (1) to (3) above, rules q_2, q_7 and q_8 also share different concatenations of these features such as (1)+(2) or (1)+(2)+(3) or (1)+(3), where + denotes the concatenation of two features. These combinations allow to better capture the similarities between rules like q_2, q_7 and q_8 and distinguish them from rules such as q_4 or q_5 which share completely different features. A key advantage of combined features is that they allow to distinguish rules from others that share only a subset of their features.

We perform feature combination by taking the cross product of all features in our training examples. This generates a very large number of features. In order to deal with those, we use two functions of Vowpal Wabbit: feature hashing and quadratic feature expansion. Feature hashing [Weinberger et al., 2009] is important for scaling the classifier to the enormous number of features created by the cross-product expansion. The quadratic expansion allows us to take the cross-product of features without having to actually write this expansion to disk, which would be prohibitive.

[6]These features have already been presented above. For better readability, we present them again here.

6.6 Evaluation

We present an evaluation of our rule selection model. We first present the setup of our experiment before discussing the systems we compare. Finally, we discuss the obtained results.

6.6.1 Experimental Setup

We evaluate our rule selection model on three domains which we denote by (i) `news`, (ii) `medical`, and (iii) `science`. The training data for `news` is from the Europarl-v4 corpus. Development and test sets are taken from the news translation task of WMT 2009 [Callison-Burch et al., 2009]. For `medical` we use the biomedical data from EMEA [Tiedemann, 2009]. Because EMEA is a parallel corpus only, we had to further process it to construct training and evaluation data. To this aim, we first removed duplicate sentences from the parallel corpus and then randomly removed sentence pairs to build the development and test data. The training data for `science` is the scientific abstracts data provided by [Carpuat et al., 2013]. We give an overview of the corpora sizes in Table 6.2.

The translation model is trained in the standard way by computing word alignments with GIZA++ [Och and Ney, 2003]. After training we reduced the size of the obtained grammar with significance testing [Johnson et al., 2007]. For feature extraction, we parsed the French part of the training data with the Berkeley parser [Petrov et al., 2006]. Lemmas and POS tags were obtained using Morfette [Chrupała et al., 2008]. We trained our rule selection model with Vowpal Wabbit[7]. For tuning, we used batch MIRA [Cherry and Foster, 2012]. We measured translation quality with 4-gram BLEU [Papineni et al., 2002], which we computed on tokenized and lowercased data. The pairwise bootstrap resampling technique of [Koehn, 2004] was used to compute statistical significance.

[7] `http://hunch.net/~vw/`. Implemented by John Langford and others.

	news	medical	science
training data	4th EuroParl corpus	[Tiedemann, 2009]	[Carpuat et al., 2013]
training data size	149,986 sentence pairs	111,081 sentence pairs	139,199 sentence pairs
development size	1,025 sentences	2,000 sentences	2,907 sentences
test size	1,026 sentences	1,999 sentences	3,915 sentences

Table 6.2: Overview of the sizes of the three domains.

6.6.2 Compared Systems

We compare our global rule selection model against three different baselines:

(i) A hierarchical system with standard parameters as in [Chiang, 2005].

(ii) A hierarchical system augmented with a local rule selection model as in [He et al., 2008, He et al., 2010]

(iii) A hierarchical system augmented with a pruned rule selection model as in [Cui et al., 2010].

The three evaluations are presented in Tables 6.3, 6.4 and 6.5 in the next section. Our global rule selection model as well as its local variants are trained using the feature templates presented in Section 6.2.2. We combine the templates into three setups:

1. Rule shape and lexical features (denoted by Lexical)
2. Rule shape and syntactic features (denoted by Syntactic)
3. Rule shape, lexical and syntactic features (denoted by Lexical and Syntactic)

We constructed our third baseline (system augmented with pruned model) by training a global rule selection model and heavily pruning out negative samples. More precisely, we reproduced the context-based target model in [Cui et al., 2010], presented in Section 3.2, by pruning as many negative examples as required to obtain approximately the same number of positive and negative samples they report. To this aim, we removed negative instances created from rules with target side frequency < 5000. For this third evaluation, we used the setup that performed best with global models, that is syntactic features only (see Section 6.6.3).

6.6.3 Results

Our first evaluation compares our global rule selection model in all setups against a hierarchical system with standard features. The results are given in Table 6.3. These show that on `science`, our global rule selection model outperforms the hierarchical baseline in all setups. Among these, the model including syntactic features only performs best. On `medical` all setups outperform the baseline but only the model with syntactic features yields significant improvements. The good results obtained by systems with syntactic features only are due to the better generalization of these, which are less sparse than the lexical ones. This ability to generalize is especially important in a global model with feature combination. Our experiments show that even a combination of syntactic and lexical features underperforms syntactic features alone because of the sparse lexical features. Our evaluation also shows that on `news` no improvements were observed. This result can be explained by the fact that `news` is much more heterogeneous than `science` or `medical`. For this task, a rule selection model should be trained on more data than our framework can handle.

Our second evaluation compares global rule selection models against local ones. The results are given in Table 6.4. The evaluation shows that on `science` and `medical`, global models outperform their local variants for all feature templates. However, on `science` only the system with syntactic features yields significant improvements. These results show that the feature sharing enabled by global models (see Section 6.5.1) yields improvements in translation quality. The results also show that global models are especially helpful when trained with syntactic features only. On news, global models do no perform better than local ones. This result is again caused by the heterogeneity of the `news` data.

Our third evaluation compares our global model with a pruned version. The results are given in Table 6.5. On `science` and `medical` the global model without pruning significantly outperforms its pruned variant. This shows that pruning hurts translation quality. On `news` no difference is observed which is due to the fact that in general rule selection models do not help on this heterogeneous domain.

System	Science	Medical	News
Hierarchical	31.22	48.67	17.28
Global lexical (1)	**31.69**	48.94	16.89
Global syntactic (2)	**32.27**	**49.66**	17.38
Global lexical and syntactic (3)	**31.89**	48.97	17.26

Table 6.3: Evaluation results. The results in bold are statistically significant improvements over the Baseline (at confidence $p < 0.05$).

System	Science	Medical	News
Hierarchical	31.22	48.67	17.28
Local lexical (1)	31.50	48.43	17.08
Local syntactic (2)	**31.85**	48.76	17.50
Local lexical and syntactic (3)	**31.74**	48.51	17.30
Global lexical (1)	**31.69**	*48.94	16.89
Global syntactic (2)	***32.27**	***49.66**	17.38
Global lexical and syntactic (3)	**31.89**	*48.97	17.26

Table 6.4: Evaluation results. The results in bold are statistically significant improvements (at confidence $p < 0.05$) over the hierarchical baseline. We use * to mark global systems that yield significant improvements over the local variants.

6.7 Conclusion

We have presented a discriminative rule selection model to improve hierarchical SMT. In contrast to previous work on this topic [He et al., 2008, He et al., 2010, Cui et al., 2010], we defined a model that (i) is global and (ii) does not prune out negative instances before training. After a brief overview of our approach, we presented our global rule selection model and showed its advantages over local and pruned models. These advantages have been confirmed by an extensive empirical evaluation which yielded up to 1 BLEU improvements over a hierarchical baseline.

A shortcoming of this work is that our approach does not scale to training data containing more than 300,000 sentence pairs. An important piece of future work is to improve our classification framework to scale to the size of public evaluation

System	Science	Medical	News
Hierarchical	31.22	48.67	17.28
Global syntactic (2)	*__32.27__	*__49.66__	17.38
Global syntactic pruned	31.00	48.61	17.18

Table 6.5: Evaluation results. The results in bold are statistically significant improvements (at confidence $p < 0.05$) over the hierarchical baseline. We use * to mark global systems that yield significant improvements over the pruned variant.

campaigns such as [Bojar et al., 2014]. Another direction would be to use a more fine-grained set of syntactic features.

Chapter 7

Improved Rule Selection for String-to-Tree Machine Translation

We present our second contribution to the topic of syntax-based SMT with soft syntactic constraints, that is the definition of a global rule selection model for string-to-tree SMT. This work is an extension of the rule selection model presented in Chapter 6 to work on the string-to-tree system described in [Williams and Koehn, 2012] and implemented in Moses [Hoang et al., 2009]. We describe the inner workings of this system in Sections 4.3.5 and 4.4.4. A detailed overview of our contribution is given in Section 3.4.

We begin by showing how to adapt the hierarchical rule selection problem described in Section 3.3 to string-to-tree systems. Then we present our model in Section 7.2 and evaluate it in Section 7.3. We conclude this chapter by explaining shortcomings of our approach and possible future work.

7.1 String-to-Tree Rule Selection

As seen in Section 3.3, a syntactic rule selection model for string-to-tree SMT tackles the problem of selecting the rule with the correct target side among rules with

the same source side. A possible formulation of this problem would be to choose, among rules such as q_3 to q_6, the one that correctly applies to sentence F.

(q_3) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, JJ_1 characteristic $JJ_2\ \rangle$
(q_4) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, NNS_1 characteristic $JJ_2\ \rangle$
(q_5) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, JJ_1 properties $JJ_2\ \rangle$
(q_6) $X/NP \rightarrow \langle\ X_1$ caractéristiques X_2, $JJ_1\ JJ_2$ features $\rangle$

F (Diverses)$_{X1}$ caractéristiques (importantes)$_{X2}$ n'ont pas été prises en compte.
(Various)$_{X_1}$ ***characteristics*** *(important)*$_{X_2}$ *were not considered.*
E (Various)$_{JJ1}$ (important)$_{JJ_2}$ **characteristics** were not considered.

This way to consider string-to-tree rule selection relies on the assumption that competing rules during decoding are the ones with the same source side. This assumption is reasonable, as it considers those rules which (i) are applied to the same segments in the input sentence and (ii) produce the same target head labels, are competing. However, this mechanism is not fully compatible with the representation and normalization of string-to-tree rules in the Moses toolkit [Hoang et al., 2009], which is the framework used in this thesis. The string-to-tree rules encode target non-terminals in the source side of the rules and the default scoring procedure normalizes rules over the source side *and target non-terminals*. This second perspective strictly takes into account partial target trees built during decoding. For instance, when a rule is selected to translate sentence F given the partial translations in Figure 7.1, then the non-terminals in the target side of this rule must match the constituents selected so far. Following this second perspective, rules q_3 and q_4 above are not competing during rule selection because their target side non-terminals are different. Competing rules for q_3 would be q_5 and q_6.

As a first attempt to model rule selection for string-to-tree rules, we implement the second variant described above and empirically evaluate it.

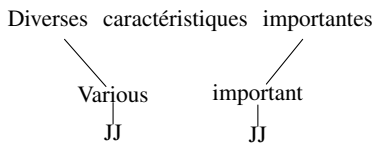


Figure 7.1: Partial translation during decoding.

7.2 Rule selection model

We denote string-to-tree rules with $X/A \rightarrow \langle \alpha, \gamma, \sim \rangle$. By $\tilde{N}t_t$, we represent the sequence of non-terminals in the target side of the rules with their alignment to non-terminals in the source side. For rules q_3, q_5 and q_6, $\tilde{N}t_t$ would be JJ_1 and JJ_2. To denote the context information of the source sentence f and the source side α of the rules, we use the same notation as for the hierarchical rule selection model in Section 6.2: $C(f, \alpha)$ is the context information in the source sentence f and the source side α. $R(\alpha, \gamma)$ are the features on the structure of the string-to-tree rules. The rule selection model estimates $P(\gamma \mid \ C(f, \alpha), R(\alpha, \gamma), \alpha, \tilde{N}t_t)$ and is normalized over the set G' of candidate target sides γ' for a given α *and* $\tilde{N}t_t$. The function $GTO : \alpha \rightarrow G'$ generates, given the source side α and target non-terminals $\tilde{N}t_t$, the set G' of all corresponding target sides γ'. The estimated distribution can be written as:

$$P(\gamma \mid C(f, \alpha), R(\alpha, \gamma), \alpha, \tilde{N}t_t) = \frac{\prod_i h_i(C(f, \alpha), R(\alpha, \gamma), \alpha, \tilde{N}t_t)^{\lambda_i}}{\sum_{\gamma' \in GTO(\alpha, \tilde{N}t_t)} \prod_i h_i(C(f, \alpha), R(\alpha, \gamma'), \alpha, \tilde{N}t_t)^{\lambda_i}}$$

In the same way as our hierarchical rule selection model, our string-to-tree model is global instead of being local to the source side of each rule. The feature templates we use are the syntactic context features and the rule shape features presented in Section 6.2.2. Our rule selection model is an additional feature in the string-to-tree translation model.

We create training examples using the shallow string-to-tree STSG procedure in

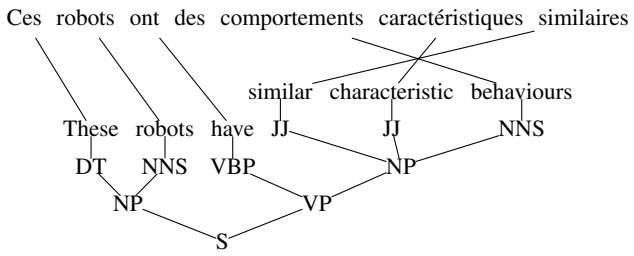


Figure 7.2: Training example for string-to-tree rule extraction.

[Williams and Koehn, 2012], which is presented in Section 4.3.5.[1] We first generate an SCFG grammar with target side annotations using this procedure. Then, each time a rule $r : X/A \rightarrow \langle \alpha, \gamma, \sim \rangle$ can be extracted from the training data, we create a new training example. In this example, the target side γ of the extracted rule is a positive instance and gets a loss of 0. To create negative samples, we search in our grammar for all rules $r_2, \ldots, r_n$ with the same source language side as r as well as the same aligned target non-terminals $\tilde{N}t_t$. Each of these constitutes a negative example and gets a cost of 1. To illustrate this example acquisition procedure, consider that rule q_3 above has been extracted from the training example in Figure 7.2. The target side "JJ_1 characteristic JJ_2 " is a correct class and gets a cost of 0. The target side of all other rules having the same source side *and aligned target non-terminals*, such as rule r_5 and r_6, are incorrect classes.

We train our string-to-tree model in the same way as for the hierarchical model (see Section 6.3): with the cost-sensitive one-against-all-reduction [Beygelzimer et al., 2005] of Vowpal Wabbit. We avoid overfitting the training data by employing early stopping once classifier accuracy decreases on a held-out dataset.

[1] Which is based on [Galley et al., 2004, Galley et al., 2006, DeNeefe et al., 2007].

7.3 Experiments

7.3.1 Experimental Setup

As our baseline system, we use the string-to-tree system implemented in Moses [Hoang et al., 2009, Williams and Koehn, 2012] with standard parameters. The implemented rule extraction [Williams and Koehn, 2012] is performed as in [Galley et al., 2004] with rule composition [Galley et al., 2006, DeNeefe et al., 2007]. Non-lexical unary rules are removed [Chung et al., 2011] and scope-3 pruning [Hopkins and Langmead, 2010] is applied on the grammar. Default rule scoring is done with relative frequencies normalized over the source side and aligned non-terminals in the target rhs. The contrastive system is the same system augmented with our rule selection model as a feature of the log-linear model.

We train and evaluate the baseline and our global model on the same data as done in the hierarchical case (see Section 6.6), that is (1) `news`, (2) `medical`, and (3) `science`. The data is processed in exactly the same way as explained in Section 6.6 except that the English side has been parsed with the Berkeley parser [Petrov et al., 2006]. Translation and language models are trained using GIZA++ [Och and Ney, 2003] and the SRI Language Modeling Toolkit [Stolcke, 2002]. After training, we reduced the number of translation rules by only keeping the 30-best rules with the same source side according to the direct rule translation probability. The rule selection model was trained with Vowpal Wabbit[2] and tuning was done with batch MIRA [Cherry and Foster, 2012]. We measured the overall translation quality with 4-gram BLEU [Papineni et al., 2002], which was computed on tokenized and lowercased data for all systems. Statistical significance is computed with the pairwise bootstrap resampling technique of [Koehn, 2004].

[2] `http://hunch.net/~vw/`

System	science	medical	news
Baseline	34.06	49.87	18.35
Contrastive	34.36	49.57	18.59

Table 7.1: String-to-tree system evaluation results.

7.3.2 Results

Table 7.1 shows the BLEU scores obtained by the evaluated systems: On `science` and `news`, small improvements are observed while for `medical` a small decrease is observed. None of these differences is statistically significant.

An analysis of the system outputs for each domain shows that the lack of significant improvements is caused by the lack of diversity between competing rules. The amount of competing rules can be estimated by counting the negative samples collected for each training example. This analysis shows that the diversity of rules containing non-terminal symbols is limited. For instance, rules s_1 to s_3 below are the only competing rules with source side *à* X_1 X_2 *éventail* X_3.

(s_1) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 variety PP_3 $\rangle$
(s_2) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 range PP_3 $\rangle$
(s_3) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 array PP_3 $\rangle$

This number is very low given that the rules contain three non-terminal symbols out of which two are adjacent. Moreover, the difference between s_1 to s_3 is limited to the lexical translation of *éventail*. Clearly, this low diversity is caused by the constraint that competing string-to-tree rules must have the same aligned non-terminal symbols. In other words, the ambiguity between translation rules in a string-to-tree system is heavily restricted by the target side syntax. Better diversity could be obtained by allowing rules with the same source side to have different aligned target non-terminals, i.e. by implementing option (1) presented in Section 7.1. This would require to normalize rule counts over the source side only as in [Williams and Koehn, 2012]. The rule selection model in Section 7.2 should then be redefined and normalized over all rules with the same source side.

A comparison between the scores obtained by the evaluated string-to-tree systems and the hierarchical system evaluation in Section 6.6 shows that although rule selection models significantly improve hierarchical machine translation, the BLEU scores obtained by these systems are significantly lower than those of the string-to-tree systems. This shows that systems with target side syntax better disambiguate than hierarchical models with rule selection. An interesting extension to the work in this thesis would be to implement string-to-tree rule selection with a normalization over all rules with the same source side.

7.4 Conclusion

We have extended the discriminative rule selection model presented in Chapter 6 to work on string-to-tree systems. Our evaluation has shown that although rule selection significantly improves hierarchical SMT, no improvement is observed on string-to-tree systems.

A shortcoming of our contribution is the requirement that only rules with the same aligned target non-terminals are considered as competing during decoding (see Section 7.1). In future work, the string-to-tree rule selection model could be redefined to consider all rules with the same source side as competing, independently of the target non-terminals.

Chapter 8

Conclusion and Future Work

We begin by discussing again the contributions listed in Section 1.2 in the light of their presentation in Chapters 5, 6 and 7. Then we discuss the shortcomings of our approaches and future work.

8.1 Contributions

We have built the first SMT system based on the Shallow Local Multi Bottom Up Tree Transducer (Sh-l-MBOT). This effort required the definition of a translation model specific to Sh-l-MBOT and the implementation of a customized decoding procedure. Our system works for Sh-l-MBOT without syntactic annotations or with annotations at different levels. An extensive evaluation has shown that Sh-l-MBOT outperform SCFG-based systems when they carry syntactic annotations on the target sides of the translation rules.

Through the integration of a high-speed classifier in the Moses toolkit, we have been able to define a rule selection model that is global, i.e. trained over the complete set of hierarchical rules, and that performs no pruning. We compared our system to previous work, which either defines models that are local to the source side of the rules or that heavily prune the training data. We showed that our global

model significantly outperforms local or pruned approaches by keeping important information for model training.

We applied our global rule selection model to approaches using syntactic annotations on the target language side (string-to-tree systems). We showed that in order to work with these models, the rule selection problem had to be reformulated. A preliminary evaluation has shown that models with improved rule selection do not outperform standard string-to-tree systems. A detailed analysis revealed that the lack of improvement has two causes : (i) there is not enough diversity in competing rules and (ii) the target side syntactic labels already perform the desired disambiguation.

Finally, we contributed to research in SMT by making our systems publicly available. The released code allows other researchers to replicate our experiments and to improve on our work.

8.2 Shortcomings and Future Work

In this section, we present shortcomings of our work and discuss how to improve on it in future work.

8.2.1 Language Model Scoring in Sh-l-MBOT Decoding

As discussed in Section 5, the major shortcoming of our Sh-l-MBOT system is the computation of Language Model (LM) scores. To illustrate this weakness, we show again the LM computation for the sentence *They predicted an increase*. Figure 8.1 shows LM scoring when decoding the word *predicted*. The complete decoding process is illustrated in Figure 5.20 (Section 5.3.3). The function $P_{LM}(w)$ computes a 4-gram LM for the (contiguous) sequence w. For better illustration, we display in red at the top of each item the rule from which it was created.

Figure 8.1 shows that for the discontiguous translation options *sind; von; ausge-*

	X $\xrightarrow{0.6}$ ⟨ predicted, sind ; von; ausgegangen ⟩ $[X \rightarrow$ predicted•$, 1, 2]$ [sind ; von ; ausgegangen$, 0.6, 3]$ $[P_{LM}$(sind)$ * P_{LM}(von) * P_{LM}$(ausgegangen)$]$	
	X $\xrightarrow{0.3}$ ⟨ predicted, haben ; vorausgesagt ⟩] $[X \rightarrow$ predicted•$, 1, 2]$ [haben ; vorausgesagt$, 0.3, 2]$ $[P_{LM}$(haben)$ * P_{LM}$(vorausgesagt)$]$	
	X $\xrightarrow{0.1}$ ⟨ predicted, prognostizierten ⟩ $[X \rightarrow$ predicted•$, 1, 2]$ [prognostizierten$, 0.1, 0, 1]$ $[P_{LM}$(prognostizierten)$]$	
They	*predicted*	*an increase*
0 1	1 2	2 4

Figure 8.1: Language model scoring of translation options for span 1-2

gangen and *haben; vorausgesagt*, the 4-gram LM is applied to each (discontiguous) component of the translation model. The computed scores are then multiplied. In our example, this comes down to a multiplication of independent unigrams, which is an imprecise approximation of the LM scores for the verbal complexes *sind; von; ausgegangen* and *haben; vorausgesagt*. In general, our scoring strategy produces lower scores for discontiguous units. For instance the sequence *sind; von; ausgegangen* gets a much lower score than *prognostizierten* although it is an equally good translation of *predicted*. In a decoding procedure with pruning, this leads to the removal of potentially good translation options.

This problem could be avoided by using a language model that can capture dependencies between discontiguous units. Future work on decoding for Sh-l-MBOT could replace our scoring method with a model considering the target segments as being dependent of each other.

8.2.2 Finer-grained Syntactic Features for Rule Selection

The rule selection models presented in Chapters 6 and 7 can be improved by defining a finer-grained set of syntactic features. The features of the rule selection model presented in Section 6.2.2 can be summarized as:

1. The matching or crossing of a syntactic constituent
2. The type of the matched constituent
3. The parent of the matched constituent
4. The lowest parent of an unmatched constituent
5. The span width covered by the rule

To illustrate these features, consider that rule q_1 is applied to sentence F_1 in Figure 8.2.

q_1 X $\xrightarrow{0.2}$ $\langle$ X_1 pratique X_2, practical X_1 X_2 $\rangle$

In our rule selection model, the syntactic features for this example are:

1. There is no matching of a syntactic constituent
2. There is no type of matched constituent
3. There is no parent of matched constituent
4. The label of the lowest parent is NP
5. The span covered by the rule has width 5

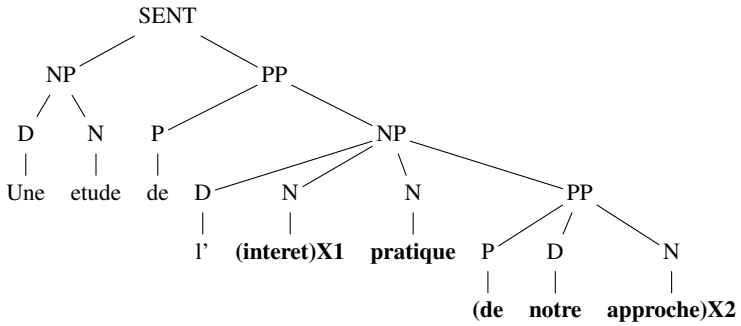


Figure 8.2: Parse tree of Sentence F_1

These features could be more fine-grained by using a tagset similar to Combinatory Categorial Grammars (CCG) [Steedman, 1996]. For instance, instead of simply

indicating a constituent mismatch (feature (1)), further features could capture the type of segment that is missing. In our example, the determiner D at the left of the rule application is missing to match a NP. Two additional features could be used to indicate (i) the type of the missing constituent and (ii) its position relative to the lowest parent. For instance, a feature D/NP would indicate that the constituent D at the left of rule application is missing to match a NP. Based on this information, further features can be defined such as the span of the missing constituent.

Future work on hierarchical rule selection could extend our approach to include such features and empirically compare this new system to our rule selection model.

8.2.3 Improved Diversity in String-to-Tree Rule Selection

Our rule selection model for string-to-tree SMT did not improve over a string-to-tree baseline (see Section 7.3). An analysis of competing rules during decoding has shown that one reason for this lack of improvement could be that there are too few competing rules. As an example, we have shown that the only competing rules with source side *à* X_1 X_2 *éventail* X_3 found in our training data are:

(s_1) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 variety PP_3 $\rangle$
(s_2) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 range PP_3 $\rangle$
(s_3) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to DT_1 JJ_2 array PP_3 $\rangle$

Given that the source side of s_1 to s_3 contains adjacent non-terminals, this number is very low. The main reason for that is that our rule selection model considers rules with the same source side *and aligned non-terminals* as being competing. As shown in Section 7.1, this restriction was necessary to make our model compatible with the default string-to-tree rule scoring procedure implemented in Moses. By dropping the requirement that aligned target non-terminals in candidate rules must be the same, rule diversity can be improved. For instance, rules t_1 to t_3 would be competing with s_1 to s_3.

(t_1) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to JJ_1 JJ_2 variety PP_3 $\rangle$

(t_2) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to NN_1 JJ_2 range $PP_3 \rangle$
(t_3) $X/PP \rightarrow \langle$ à X_1 X_2 éventail X_3, to JJ_1 NN_2 array $NP_3 \rangle$

Future work on string-to-tree rule selection could redefine our rule selection model by dropping the constraint on aligned target non-terminals. To achieve that, the default rule scoring procedure should be replaced by one that normalizes rules over the source side only as in [Williams and Koehn, 2012]. An empirical evaluation could compare the new model to ours.

8.2.4 Scalability of Rule Selection Models

We have mentioned in Section 6.7 that a shortcoming of our rule selection models is that they do not scale to training data containing more than 300,000 sentence pairs. This bottleneck prevents us to train systems on enough data to participate in public evaluation campaigns such as [Bojar et al., 2014]. An important future work would be to improve our framework to scale to the size of the training data provided in shared tasks.

8.2.5 Rule Selection for Sh-l-MBOT

In this thesis, we have applied rule selection to hierarchical and string-to-tree SMT. An interesting research direction would be to define rule selection models for Sh-l-MBOT systems. Because Sh-l-MBOT rules allow target side discontiguities, there are very large amounts of rules with the same source side (which are thus competing during decoding). This makes Sh-l-MBOT systems very good candidates for rule selection models. Moreover, as these systems work with syntactic annotations at any level, rule selection could be tested in different settings such as hierarchical or string-to-tree.

Bibliography

[Aho and Ullman, 1969] Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*

[Almaghout et al., 2011] Almaghout, H., Jiang, J., and Way, A. (2011). CCG contextual labels in hierarchical phrase-based SMT. In *Proc. EAMT*.

[Ambati and Lavie, 2008] Ambati, V. and Lavie, A. (2008). Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proc. AMTA*.

[Ambati et al., 2009] Ambati, V., Lavie, A., and Carbonell, J. (2009). Extraction of syntactic translation models from parallel data using syntax from source and target languages. In *Proc. MT Summit*.

[Baroni et al., 2009] Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*.

[Beygelzimer et al., 2005] Beygelzimer, A., Langford, J., and Zadrozny, B. (2005). Weighted one-against-all. In *Proc. AAAI*.

[Blunsom et al., 2009] Blunsom, P., Cohn, T., Dyer, C., and Osborne, M. (2009). A Gibbs sampler for phrasal synchronous grammar induction. In *Proc. ACL*.

[Blunsom et al., 2008] Blunsom, P., Cohn, T., and Osborne, M. (2008). A discriminative latent variable model for statistical machine translation. In *Proc. ACL*.

[Bojar et al., 2013] Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. WMT*.

[Bojar et al., 2014] Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 Workshop on Statistical Machine Translation. In *Proc. WMT*.

[Braune et al., 2013] Braune, F., Maletti, A., Quernheim, D., and Seemann, N. (2013). Shallow local multi bottom-up tree transducers in statistical machine translation. In *Proc. ACL*.

[Brown et al., 1990] Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Comput. Linguist.*

[Brown et al., 1993] Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*

[Buckwalter, 2002] Buckwalter, T. (2002). Arabic transliteration. `http://www.qamus.org/transliteration.htm`.

[Callison-Burch et al., 2009] Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 workshop on statistical machine translation. In *Proc. WMT*.

[Carpuat et al., 2013] Carpuat, M., Daume III, H., Henry, K., Irvine, A., Jagarlamudi, J., and Rudinger, R. (2013). Sensespotting: Never let your parallel data tie you to an old domain. In *Proc. ACL*.

[Carpuat and Wu, 2007] Carpuat, M. and Wu, D. (2007). Improving statistical machine translation using word sense disambiguation. In *Proc. EMNLP*.

[Chan et al., 2007] Chan, Y. S., Ng, H. T., and Chiang, D. (2007). Word sense disambiguation improves statistical machine translation. In *Proc. ACL*.

[Chappelier et al., 1998] Chappelier, J.-C., Rajman, M., et al. (1998). A generalized CYK algorithm for parsing stochastic CFG. *TAPD*.

[Charniak and Johnson, 2005] Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. ACL*.

[Cherry and Foster, 2012] Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proc. NAACL*.

[Chiang, 2005] Chiang, D. (2005). Hierarchical phrase-based translation. In *Proc. ACL*.

[Chiang, 2006] Chiang, D. (2006). An introduction to synchronous grammars.

[Chiang, 2007] Chiang, D. (2007). Hierarchical phrase-based translation. *Comput. Linguist.*

[Chiang, 2010] Chiang, D. (2010). Learning to translate with source and target syntax. In *Proc. ACL*.

[Chrupała et al., 2008] Chrupała, G., Dinu, G., and Van Genabith, J. (2008). Learning morphology with morfette. In *Proc. LREC*.

[Chung et al., 2011] Chung, T., Fang, L., and Gildea, D. (2011). Issues concerning decoding with synchronous context-free grammars. In *Proc. ACL*.

[Cohn and Blunsom, 2009] Cohn, T. and Blunsom, P. (2009). A Bayesian model of syntax-directed tree to string grammar induction. In *Proc. EMNLP*.

[Cui et al., 2010] Cui, L., Zhang, D., Li, M., Zhou, M., and Zhao, T. (2010). A joint rule selection model for hierarchical phrase-based translation. In *Proc. ACL*.

[DeNeefe and Knight, 2009] DeNeefe, S. and Knight, K. (2009). Synchronous tree adjoining machine translation. In *Proc. EMNLP*.

[DeNeefe et al., 2007] DeNeefe, S., Knight, K., Wang, W., and Marcu, D. (2007). What can syntax-based MT learn from phrase-based MT. In *Proc. EMNLP*.

[Ding and Palmer, 2005] Ding, Y. and Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. ACL*.

[Eisele and Chen, 2010] Eisele, A. and Chen, Y. (2010). MultiUN: A multilingual corpus from United Nation documents. In *Proc. LREC*.

[Eisner, 2003] Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proc. ACL*.

[Fossum et al., 2008] Fossum, V., Knight, K., and Abney, S. (2008). Using syntax to improve word alignment precision for syntax-based machine translation. In *Proc. WMT*.

[Galley et al., 2006] Galley, M., Graehl, J., Knight, K., Marcu, D., Deneefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL*.

[Galley et al., 2004] Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proc. NAACL*.

[Gao et al., 2011] Gao, Y., Koehn, P., and Birch, A. (2011). Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proc. EMNLP*.

[Graehl and Knight, 2004] Graehl, J. and Knight, K. (2004). Training tree transducers. In *Proc. HLT-NAACL*.

[Graehl et al., 2008] Graehl, J., Knight, K., and May, J. (2008). Training tree transducers. *Comput. Linguist.*

[Habash et al., 2009] Habash, N., Rambow, O., and Roth, R. (2009). MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proc. MEDAR*.

[Hanneman and Lavie, 2013] Hanneman, G. and Lavie, A. (2013). Improving syntax-augmented machine translation by coarsening the label set. In *Proc. NAACL-HLT*.

[He et al., 2008] He, Z., Liu, Q., and Lin, S. (2008). Improving statistical machine translation using lexicalized rule selection. In *Proc. COLING*.

[He et al., 2010] He, Z., Meng, Y., and Yu, H. (2010). Maximum entropy based phrase reordering for hierarchical phrase-based translation. In *Proc. EMNLP*.

[Hoang, 2011] Hoang, H. (2011). *Improving statistical machine translation with linguistic information*. PhD Thesis: The University of Edinburgh.

[Hoang and Koehn, 2010] Hoang, H. and Koehn, P. (2010). Improved translation with source syntax labels. In *Proc. WMT*.

[Hoang et al., 2009] Hoang, H., Koehn, P., and Lopez, A. (2009). A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proc. IWSLT*.

[Hopcroft et al., 2006] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc.

[Hopkins and Langmead, 2010] Hopkins, M. and Langmead, G. (2010). SCFG decoding without binarization. In *Proc. EMNLP*.

[Huang et al., 2006] Huang, L., Knight, K., and Joshi, A. (2006). A syntax-directed translator with extended domain of locality. In *Proc. AMTA*.

[Huang et al., 2009] Huang, L., Zhang, H., Gildea, D., and Knight, K. (2009). Binarization of synchronous context-free grammars. *Comput. Linguist.*

[Huang et al., 2010] Huang, Z., Čmejrek, M., and Zhou, B. (2010). Soft syntactic constraints for hierarchical phrase-based translation using latent syntactic distributions. In *Proc. EMNLP*.

[Johnson et al., 2007] Johnson, H., Martin, J., Foster, G., and Kuhn, R. (2007). Improving translation quality by discarding most of the phrasetable. In *Proc. EMNLP*.

[Koehn, 2004] Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

[Koehn, 2005] Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit*.

[Koehn et al., 2005] Koehn, P., Axelrod, A., Mayne, A. B., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh system description for the 2005 IWSLT Speech Translation Evaluation. In *Proc. IWSLT*.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. ACL: Demo*.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. ACL*.

[Lavie, 2008] Lavie, A. (2008). Stat-xfer: a general search-based syntax-driven framework for machine translation. In *Proc. CICLing*.

[Lavie et al., 2008] Lavie, A., Parlikar, A., and Ambati, V. (2008). Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proc. SSST*.

[Li et al., 2014] Li, J., Marton, Y., Resnik, P., and Daumé III, H. (2014). A unified model for soft linguistic reordering constraints in statistical machine translation. In *Proc. ACL*.

[Li et al., 2009] Li, Z., Callison-Burch, C., Dyer, C., Ganitkevitch, J., Khudanpur, S., Schwartz, L., Thornton, W. N. G., Weese, J., and Zaidan, O. F. (2009). Joshua: An open source toolkit for parsing-based machine translation. In *Proc. WMT*.

[Liu et al., 2011a] Liu, L., Zhao, T., Wang, C., and Cao, H. (2011a). A unified and discriminative soft syntactic constraint model for hierarchical phrase-based translation. In *Proc. MT Summit*.

[Liu et al., 2008] Liu, Q., He, Z., Liu, Y., and Lin, S. (2008). Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proc. EMNLP*.

[Liu et al., 2007] Liu, Y., Huang, Y., Liu, Q., and Lin, S. (2007). Forest-to-string statistical translation rules. In *Proc. ACL*.

[Liu et al., 2006] Liu, Y., Liu, Q., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proc. ACL*.

[Liu et al., 2011b] Liu, Y., Liu, Q., and Lü, Y. (2011b). Adjoining tree-to-string translation. In *Proc. ACL*.

[Liu et al., 2009] Liu, Y., Lü, Y., and Liu, Q. (2009). Improving tree-to-tree translation with packed forests. In *Proc. ACL*.

[Maletti, 2010] Maletti, A. (2010). Why synchronous tree substitution grammars? In *Proc. NAACL*.

[Maletti, 2011] Maletti, A. (2011). How to train your multi bottom-up tree transducer. In *Proc. ACL*.

[Marcu et al., 2006] Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*.

[Marton et al., 2012] Marton, Y., Chiang, D., and Resnik, P. (2012). Soft syntactic constraints for Arabic—English hierarchical phrase-based translation. *Machine Translation*.

[Marton and Resnik, 2008] Marton, Y. and Resnik, P. (2008). Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. ACL*.

[May et al., 2010] May, J., Knight, K., and Vogler, H. (2010). Efficient inference through cascades of weighted tree transducers. In *Proc. ACL*.

[Melamed, 2004] Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proc. ACL*.

[Mi and Huang, 2008] Mi, H. and Huang, L. (2008). Forest-based translation rule extraction. In *Proc. EMNLP*.

[Mi et al., 2008] Mi, H., Huang, L., and Liu, Q. (2008). Forest-based translation. In *Proc. ACL*.

[Mylonakis and Sima'an, 2010] Mylonakis, M. and Sima'an, K. (2010). Learning probabilistic synchronous CFGs for phrase-based translation. In *Proc. CoNLL*.

[Mylonakis and Sima'an, 2011] Mylonakis, M. and Sima'an, K. (2011). Learning hierarchical translation structure with linguistic annotations. In *Proc. ACL*.

[Nesson et al., 2006] Nesson, R., Shieber, S., and Rush, A. (2006). Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proc. AMTA*.

[NIST, 2010] NIST (2010). NIST 2002 [2003, 2005, 2008] open machine translation evaluation. Linguistic Data Consortium.

[Och, 2003] Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proc. ACL*.

[Och and Ney, 2002] Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL*.

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Comput. Linguist.*

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*.

[Petrov et al., 2006] Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proc. ACL*.

[Quirk et al., 2005] Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. ACL*.

[Riezler and Maxwell III, 2006] Riezler, S. and Maxwell III, J. T. (2006). Grammatical machine translation. In *Proc. NAACL-HLT*.

[Satta and Peserico, 2005] Satta, G. and Peserico, E. (2005). Some computational complexity results for synchronous context-free grammars. In *Proc. EMNLP*.

[Schmid, 2004] Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proc. COLING*.

[Seemann et al., 2015a] Seemann, N., Braune, F., and Maletti, A. (2015a). String-to-tree multi bottom-up tree transducers. In *Proc. ACL*.

[Seemann et al., 2015b] Seemann, N., Braune, F., and Maletti, A. (2015b). A systematic evaluation of MBOT in statistical machine translation. In *Proc. 15th MT Summit*.

[Setiawan et al., 2009] Setiawan, H., Kan, M.-Y., Li, H., and Resnik, P. (2009). Topological ordering of function words in hierarchical phrase-based translation. In *Proc. ACL*.

[Simianer et al., 2012] Simianer, P., Riezler, S., and Dyer, C. (2012). Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proc. ACL*.

[Steedman, 1996] Steedman, M. (1996). *Surface structure and interpretation*. MIT Press.

[Stolcke, 2002] Stolcke, A. (2002). SRILM — an extensible language modeling toolkit. In *Proc. INTERSPEECH*.

[Sun et al., 2009] Sun, J., Zhang, M., and Tan, C. L. (2009). A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proc. ACL*.

[Tamchyna et al., 2014] Tamchyna, A., Braune, F., Fraser, A., Carpuat, M., Daume III, H., and Quirk, C. (2014). Integrating a discriminative classifier into phrase-based and hierarchical decoding. In *The Prague Bulletin of Mathematical Linguistics*.

[Tiedemann, 2009] Tiedemann, J. (2009). News from OPUS : A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing V*, volume V. John Benjamins.

[Vilar et al., 2012] Vilar, D., Stein, D., Huck, M., and Ney, H. (2012). Jane: An advanced freely available hierarchical machine translation toolkit. *Machine Translation*.

[Vilar et al., 2008] Vilar, D., Stein, D., and Ney, H. (2008). Analysing soft syntax features and heuristics for hierarchical phrase based machine translation. In *Proc. IWSLT*.

[Wang et al., 2007] Wang, W., Knight, K., and Marcu, D. (2007). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP*.

[Wang et al., 2010] Wang, W., Knight, K., May, J., and Marcu, D. (2010). Restructuring, re-labeling, and re-aligning for syntax-based machine translation. In *Comput. Linguist.*

[Web-as-Corpus Consortium, 2008] Web-as-Corpus Consortium (2008). SDeWaC — a 0.88 billion word corpus for German. Website: `http://wacky.sslmit.unibo.it/doku.php`.

[Weinberger et al., 2009] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proc. ICML*.

[Williams and Koehn, 2012] Williams, P. and Koehn, P. (2012). GHKM rule extraction and scope-3 parsing in Moses. In *Proc. WMT*.

[Wu, 1997] Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*

[Xiao et al., 2009] Xiao, T., Li, M., Zhang, D., Zhu, J., and Zhou, M. (2009). Better synchronous binarization for machine translation. In *Proc. EMNLP*.

[Yamada and Knight, 2001] Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proc. ACL*.

[Yamada and Knight, 2002] Yamada, K. and Knight, K. (2002). A decoder for syntax-based statistical MT. In *Proc. ACL*.

[Zhai et al., 2013] Zhai, F., Zhang, J., Zhou, Y., and Zong, C. (2013). Handling ambiguities of bilingual predicate-argument structures for statistical machine translation. In *Proc. ACL*.

[Zhang et al., 2006] Zhang, H., Huang, L., Gildea, D., and Knight, K. (2006). Synchronous binarization for machine translation. In *Proc. NAACL-HLT*.

[Zhang et al., 2008] Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C. L., and Li, S. (2008). A tree sequence alignment-based tree-to-tree translation model. In *Proc. ACL*.

[Zhang et al., 2007] Zhang, M., Jiang, H., Aw, A. T., Sun, J., Li, S., and Tan, C. L. (2007). A tree-to-tree alignment-based model for statistical machine translation. In *Proc. MT Summit*.

[Zollmann and Venugopal, 2006] Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proc. WMT*.